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

# Final Report:
# Rocko

| | |
|---|---|
| Student Name: | Darren Kelley |
| Date: | 4/21/1999 |
| Instructor: | A. Antonio Arroyo |
| TAs: | Scott Jantz |
| | Aamir Qaiyumi |
| | Patrick O'Malley |

## Table of Contents

## Abstract

The goal of this project is to create a robot that will carry a model rocket to a launch site. Obstacle avoidance will be accomplished using infrared sensors (for inside) and 40 kHz sonar (for outside). The robot will have a 2 CDS cells on it, which it will use to determine whether it is inside or outside and to find the brightest point in the sky. A flex sensor is attached to the launch pad in order to determine the angle at which the face of the pad is positioned.

## Introduction

For my IMDL project I decided that I wanted to build a robot that would be fun. After talking with Scott Jantz I decided that a mobile rocket-launching robot is just about as fun as it gets. Model rockets are intriguing alone and if you put one on a robot that drives around and searches for the sun then you really have something fun. This project incorporated infrared emitter and detector, sonar, sonar, DC motors, photo-sensors, and a flex sensor; to accomplish the goal of making a mobile rocket-launching robot.

## Integrated System

Rocko combines standard IR emitters and detectors and sonar transmitters and a receiver to perform obstacle avoidance. Because Rocko's final objective is to launch a model rocket, it is necessary that Rock function outside. The IR sensors only detect reliably at about six inches outside, and for this reason sonar was selected as the main obstacle avoidance sensor. There is a rotational launch pad platform on top of Rocko's main platform (a toy R/C truck) to allow Rocko to select a direction to launch the rocket while stationary. Rock uses a servo to rotate the launch pad and a motor in a drive screw configuration attached to one end of the launch pad to adjust the angle. The CDS photocells give Rocko light direction feed back in order to determine the brightest point in the sky. Since the angle of the launch pad is adjusted with a motor it was necessary to incorporate a feedback system to determine the pads location. This was accomplished by placing a flex sensor under the pad so that it would be bent as the pad lowered. With all of these systems integrated Rocko can perform obstacle avoidance and then stop and find the light source and launch the rocket.

## Mobile Platform

The objective of Rocko’s platform was to take an off-the-shelf R/C truck and modify it to carry a rotational launch pad. The first problem I encountered was the front wheel steering. The R/C truck was inexpensive and did not originally incorporate a servo for steering, therefore it was necessary to modify the existing platform to be able to interface with a servo. This was accomplished by drilling two holes in the base of the R/C truck platform and placing 2-inch screws through them, on which the servo was attached.

The second problem how to mount a 10” x 14” piece of plywood on top of the platform and allow for enough room underneath the plywood to mount all of Rocko’s circuitry. This was done by using ¼” (outer diameter) pex tubing. Four standoffs were used, each was cut to a desired length in order to hold the plywood 3 inches off of the platform.

Next, the rotational launch-pad platform was constructed. This platform is a box 2 inches high and 6 inches on the sides. A height of two inches was chosen in order for the platform to house a hacked 43-oz. in. servo and the circuit boards for a sonar receiver and flex sensor. The launch-pad platform was connected to the top of the main platform using a bearing system to allow a servo (mounted on the under side of the main platform) to rotate the pad platform. At one corner of launch-pad platform a motor (hack servo) was mounted and a 4 inch 5/16” drive screw was attached to it. A 5/16” nut was then attached to one end of the launch pad. The other end of the launch pad was secured in a way that restricted its movement to forwards and backward.

The next step, and probably the most difficult, was mounting all of the sensors and circuitry.  In order to keep everything way from the area where the rocket would launch it was necessary to mount it all under the main platform.

## Actuation

Rocko incorporated two servos and two DC motors.  The two servos were 42 oz. in. servos.  One servo was used for the front wheel steering and the other to rotate the launch-pad platform.

The DC motor located in the rotational launch pad platform was a hacked 42-oz. in. servo.  This servo was incorporated into a screw drive system that moved one side of the launch pad up and down in order to change the launch angle of the rocket.  After experimenting with the screw-drive I found that the DC motor did not have a high enough top rotational speed to adjust the launch pad at a reasonable rate.

The second motor was the rear wheel drive motor that came with the R/C vehicle.  The only specification I have on this motor is that at 9.6 volts the stall current is around 8 Amps.  This is a high-speed low torque motor, and I found that on a paved surface if the vehicle were given an initial push it would continue to move with a pulse-width-modulated signal having a 15% duty cycle.  To start the vehicle from rest on a paved surface required a duty cycle of 30% or more, and on a surface such as grass a 60% duty cycle was necessary.  The problem with this was that once it broke its initial high-torque start up phase its speed would increase and it moved much faster that I would have liked.

## Sensors

### IR Sensors

The IR emitters are powered off a 40 kHz signal generated by dividing the 68HC11's E-clock by 50 with a 74HC390 decade counter, located on the ME11 board. The 40 kHz Sharp GPIU58Y IR detectors have been hacked in order to give an analog output rather than a digital one. The outputs of these detectors are connected to the A/D port, port E, on the 68HC11 EVBU.

The robot has two IR detector emitter pairs set at 90 degrees from each other. The sensors are positioned at the corners of the platform facing the center.

### Sonar Sensors

The sonar suite consists of two 40 kHz sonar transmitters, a 40 kHz sonar receiver, a 24 kHz sonar receiver, and a 24 kHz stationary sonar transmitter.

The 40 kHz sonar transmitter and receiver were constructed using the circuit diagrams provided in Megan Grimm's final report from Spring 1998. The transmitter (Figure 1) consists of an audio transformer and a 2N222A transistor. The input to the transmitter a

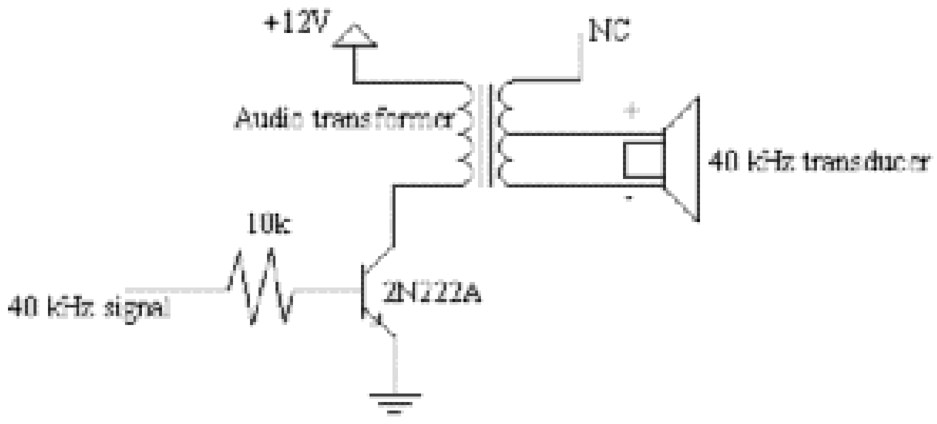


Figure 1: 40kHz Sonar Transmitter

40 kHz signal form the ME11 board (approximately .6 $V_{pp}$).

The output to a 40 kHz sonar transducer is a 40 kHz (approximately 52 $V_{pp}$) square wave. The receiver (Figure 2) consists of a MAX 266 chip (which contains 2 – second order filter) configured as a 40 kHz bandpass filter and a LM339 comparator.  The output of the comparator is feed into the analog port.

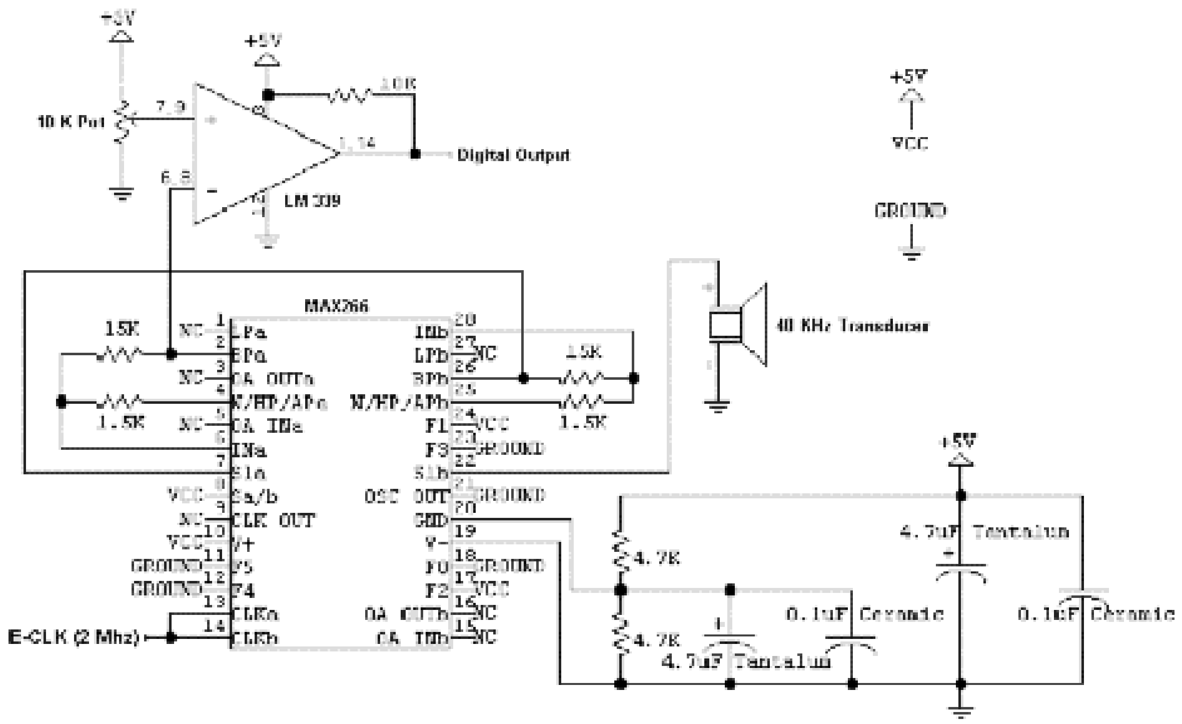


Figure 2: 40kHz Sonar Receiver Circuit

The circuit shown in Figure 3 was used to produce a 24kHz clock for the sonar

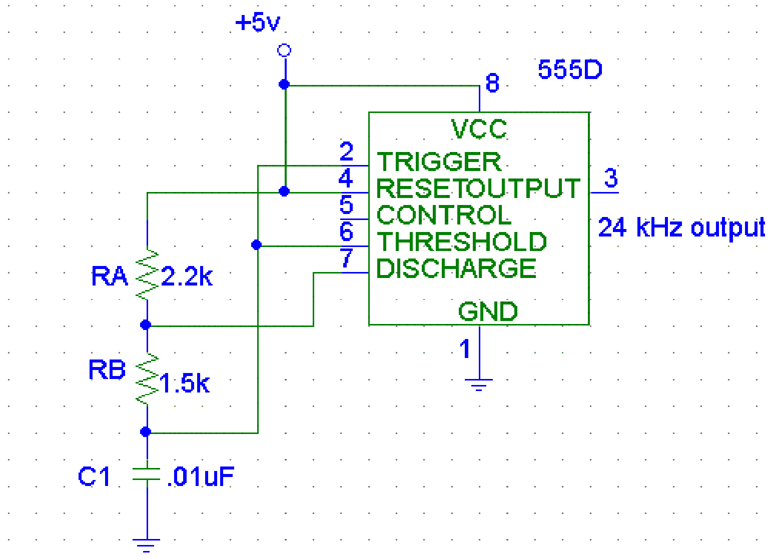
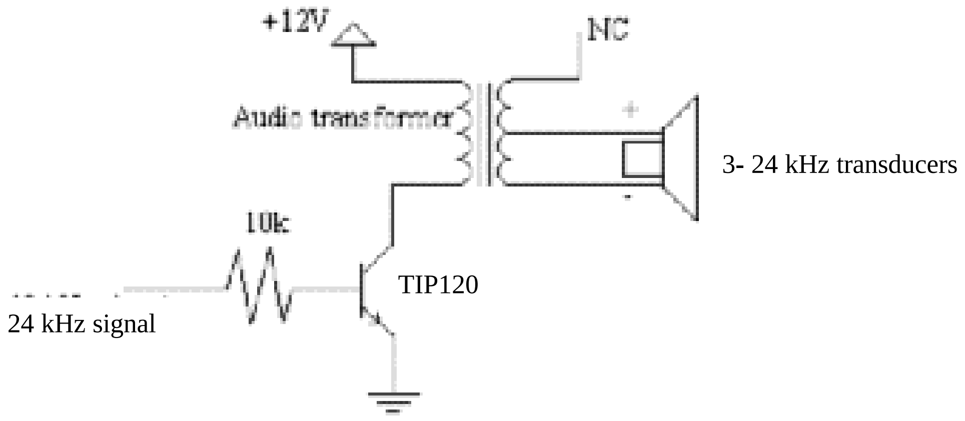


Figure 3: 24kHz Clock

transmitter circuit. The following formula was used to choose the resistor and capacitor values for a 24kHz frequency: $T = .693\ (R_A + 2R_B)\ C$. The transmitter circuit for the 24kHz sonar beacon is shown in Figure 4. Upon testing this circuit it was found to produce the desired 24kHz output. In order to use this 24kHz transmitter as a beacon it was necessary to have the 24kHz-signal input constantly. After a few minutes of testing it was found that this would cause the TIP120 to burn up, and was therefore no implemented into the final design.



The 24kHz receiver is the same at that for the 40kHz receiver with a few minor changes. There are six input pins on the MAX266, F5-F0, these pins are used to select the frequency at which the filters will operate. Figure 5 shows the correct connections in order to configure the chip for 24kHz (This is the only change to the circuit shown in Figure 2 in order to use it for 24kHz).

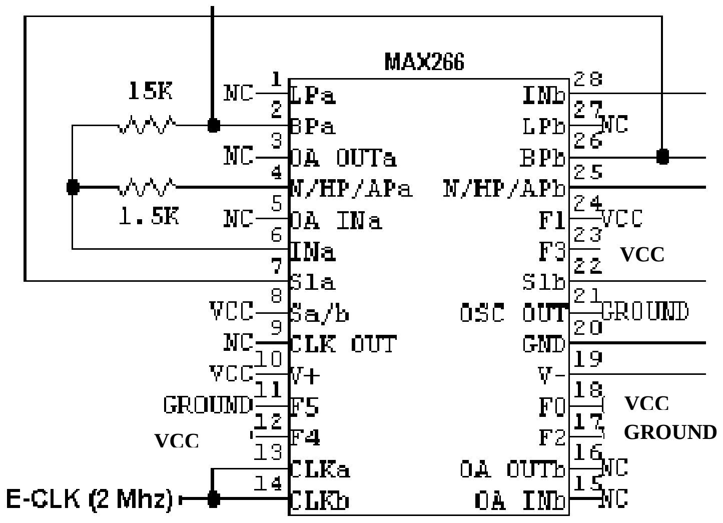


Figure 5: MAX266 Modifications for 24kHz

## Sonar Calibration

*The following information concerning calibration of the 40kHz sonar receiver was taken from the Fall '98 Ranos Final Report by William O'Connor.*

To calibrate the receivers, the emitter needs to run at a constant 40 kHz signal.

The transducers for the emitter and receiver need to face each other and even connected together. The oscilloscope probe connects to ground and the digital output of the op amp (LM339). The following figures were taken from an oscilloscope set at 5**m**s and 2 volt/div. The potentiometer needs to be adjusted by turning it right so that the voltage reading between ground and the output is around 3V (Figure 6).

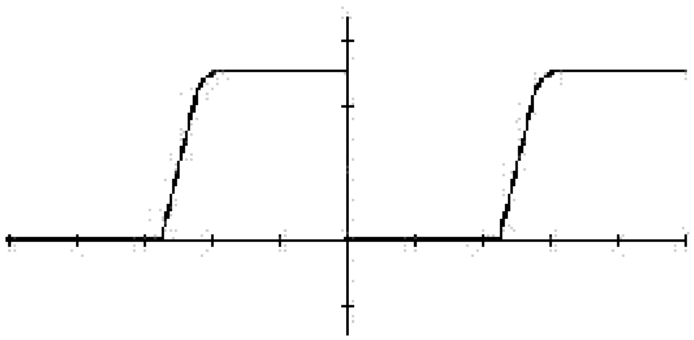


Figure 6: Correct oscilloscope output when adjusting potentiometer on sonar receiver

When turning the potentiometer to the left, the signal becomes weaker decreasing the positive value (Figure 7) until the waveform is no longer noticeable.

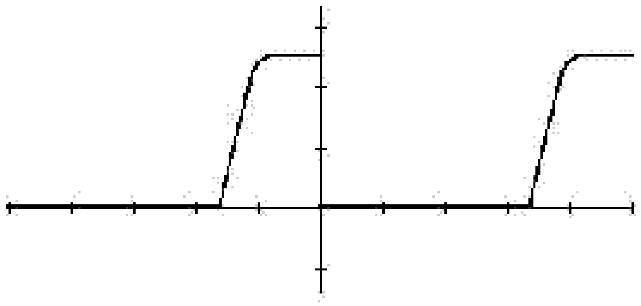


Figure 7: Example of signal from output of the comparator becoming weaker

CDS Photocells

The CDS photocells change resistance depending on the intensity of light they are exposed to. In order to detect bright light (outside where the ambient light is very high) the CDS cells were incorporated into a voltage divider circuit with a 1 kΩ resistor. To detect direction of light a CDS cell was placed at the bottom of a three sided box (open top). With this design the cell was always in a shadow unless the open side of the box was pointing in the direction of the sun. A second CDS cell was placed at the bottom of a four sided box (open top). After the direction of the sun was found this cell would rotate through a 45 degree angle and the cell would be exposed to brightest light when the angle was closest to that of the sun. The output (voltage across the CDS cell) is feed into the A/D port (port E).

Flex Sensor

This sensor is a 4-½ inch strip of bendable material that changes resistance as it bends. The nominal resistance when the sensor is unbent is 9.80 kΩ, and the sensor has a maximum resistance of 30 kΩ (when bent 180°). A voltage divider circuit was constructed using a 20 kΩ resistor and the flex sensor. An output voltage was taken across the flex sensor and feed into the A/D port (port E).

## Behaviors

Rocko has three main behaviors: obstacle avoidance, search, and launch.

### Obstacle Avoidance

IR is turned on immediately and runs continually as Rocko performs obstacle avoidance. Since two sonar transmitters, but only one sonar receiver is used there is a delay between sampling the right and left sonar.  Each is sample once every 5 ms and there is a 1 ms delay in-between their sampling (in order to prevent receiving a signal from the wrong transmitter).

### Search

The searching behavior has two phases, a horizontal-plane searching phase followed by a vertical-plane searching phase.  The horizontal-plane searching phase begins by initializing the rotational launch pad platform to its right most position.  The platform then rotates through 180 degrees taking light samples every .3 degrees and recording the largest value and the degree at which that value was found.  Once it has rotated through 180 degrees it returns to the position at which the brightest light was found.  Once in this position the same process is executed in the vertical plane, the only difference is that the position is determined by the value of the flex sensor.

### Launch

When the searching process is finished the output pin PortD bit 3 goes high and the rocket is launched.

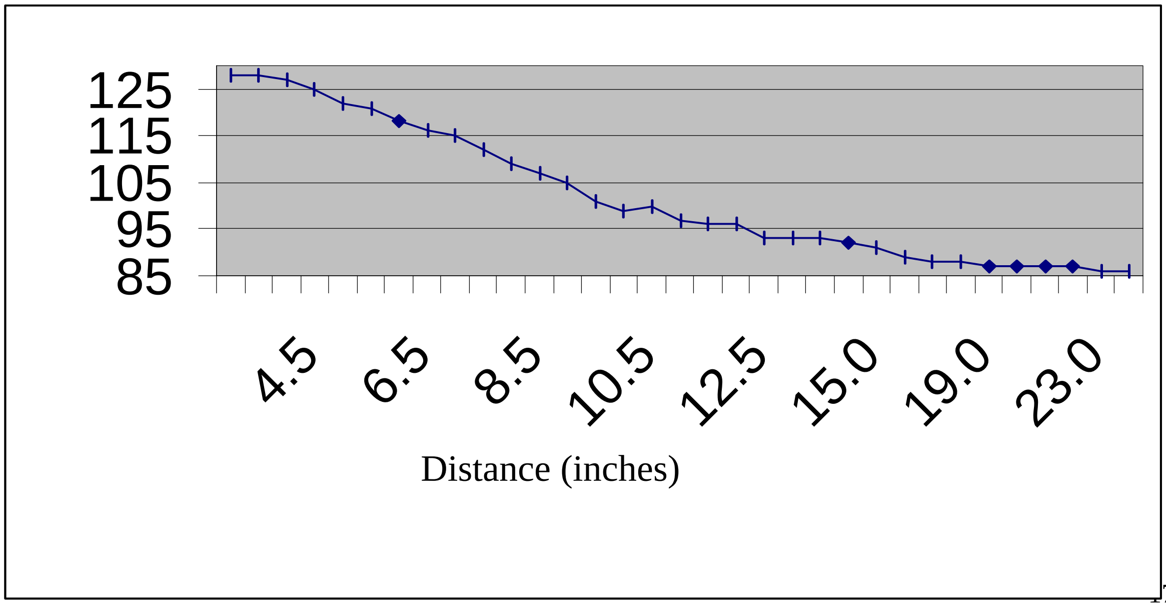
## Experiments

The experiments conducted in order to get IR, sonar, and CDS cell data were done so with bench setups (the sensors were not on the robot). The data for the flex sensor was collected from the setup, as it will be on the robot.

### Infrared Sensor

In order to determine the range and sensitivity of the IR emitter-receiver pair the following data was taken.

| Distance(inches) | IR Sensor(Port E) |
|---|---|
| 3.0 | 128 |
| 3.5 | 128 |
| 4.0 | 127 |
| 4.5 | 125 |
| 5.0 | 122 |
| 5.5 | 121 |
| 6.0 | 118 |
| 6.5 | 116 |
| 7.0 | 115 |
| 7.5 | 112 |
| 8.0 | 109 |
| 8.5 | 107 |
| 9.0 | 105 |
| 9.5 | 101 |
| 10.0 | 99 |
| 10.5 | 100 |
| 11.0 | 97 |
| 11.5 | 96 |
| 12.0 | 96 |
| 12.5 | 93 |
| 13.0 | 93 |
| 13.5 | 93 |
| 14.0 | 92 |
| 15.0 | 91 |
| 16.0 | 89 |
| 17.0 | 88 |
| 18.0 | 88 |
| 19.0 | 87 |
| 20.0 | 87 |
| 21.0 | 87 |
| 22.0 | 87 |
| 23.0 | 86 |
| 24.0 | 86 |



## Sonar Sensor

In order to determine the distance of an object from a transmitter/receiver pair a time of flight algorithm was used (this code was written by Megan Grimm). The following data illustrates the distance from the object versus the time taken for the receiver to receive a signal after the transmitter sent a pulse.

| Distance(feet) | Counter |
|---|---|
| 0.0 | 0 |
| 0.5 | 0,1 |
| 1.0 | 1,2 |
| 1.5 | 2 |
| 2.0 | 2,3 |
| 2.5 | 3,4 |
| 3.0 | 4 |
| 3.5 | 5 |
| 4.0 | 6,7 |
| 4.5 | 8 |
| 5.0 | 9,10 |

This experiment was conducted in the IMDL lab with the transmitter approximately four inches form the receiver and both approximately 3 feet off of the ground. An almost liner correlation between the distance and the time of flight counter can be seen.

## CDS Photocells

Two experiments were conducted with the CDS cells.

Experiment 1: A light was placed directly in front of the CDS cell and the cell was rotated horizontally through 45 degrees.

| Angle(deg) | CDS (Port E) |
|---|---|
| 0.00 | 135 |
| 11.25 | 173 |
| 22.50 | 233 |
| 33.75 | 248 |
| 45.00 | 251 |

Experiment 2: A light was set 2 feet off of the ground with the cell placed directly in front of it.  The cell was then rotated trough 90 degrees.

| Angle(deg) | CDS (Port E) |
|---|---|
| 0.00 | 229 |
| 11.25 | 237 |
| 22.50 | 239 |
| 33.75 | 243 |
| 45.00 | 244 |
| 56.25 | 247 |
| 67.50 | 248 |
| 90.00 | 253 |

These results show that the robot will be able to collect data for the light intensity using the CDS cells and with that data find the brightest point.

## Flex Sensor

One end of the flex sensor was attached to the bottom of the launch pad while the other was attached to the surface on which the pad is resting.  As the angle of the launch pad changed by raising one end the amount which the flex sensor was bent changed accordingly.  The following table shows the results:

| Height(inches) | Flex (Port E) |
|---|---|
| 0.0 | 135 |
| 0.5 | 139 |
| 1.0 | 151 |
| 1.5 | 149 |
| 2.0 | 146 |
| 2.5 | 142 |
| 3.0 | 136 |

## Conclusion

I was happy with the performance of Rocko's searching and launching behaviors. The sonar used for obstacle avoidance still needs a little work. It may be a timing issue and have something to do with how fast I am sampling. When prompting for a sample by hand (maybe once every second) my data was extreamly consistent. But when outputting the sonar values to the screen as my program sampled every 5 ms I saw a lot of invalid data.

I did not get to interface the 24kHz-sonar beacon with my project. The beacon itself would burn up if left on for more than a couple of minutes. I concluded that in order to use it I would have to use another controller to turn it on for 1 second and then off for 3, this would allow the TIP120 to remain at a constant temperature.

I experienced many problems with my platform. Interfacing with an already designed platform can be frustrating and time consuming. If I were to do this project over I would choose another platform, a tracked platform such at the one Justin Cobb used would have been much easier to build on and control.

## References

Valuable information and guidance was obtained through the following sources:

- The TAs: Scott Jantz, Aamir Qaiyumi , and Patrick O’Malley
- Aamir’s help in getting the version of ICC which allowed me to finish my project
- Megan Grimm’s project Alph and Ralph and her help in lab
- Willliam O’Connor’s project Ranos
- Justin Cobb’s help with sonar, debugging, and much much more

## Appendix A:

## Vendor Information

IR emitters/detectors:

- LED emitters: Mekatronics
  316 NW 17 th 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

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

R/C Truck:

- Nikko America Inc.
  2801 Summit Ave.
  Plano, TX 75074
  800-776-4556

## Appendix B: Code

```
/*********************************************************************
*
 * Title        rockofin.c
 * ProgrammerDarren Kelley
 * Date        April 15, 1999
 * Version       3
 *
 * Description
 *   This program allows my robot (rear wheel drive and front wheel steering
 *        to perform obstacle avoidance, scan for the brightest area in the sky
 *   and launch a rocket(PD3).FOR FRONT SENSORS AT INSIDE ANGLE OF 120
DEGREES
*************************************************************************
/

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

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

/************************ End of includes ******************************/

/**************************** Defines **********************************/

#define poke *(unsigned char *) (0x7000)
#define irrt analog(0)
#define irlf analog(1)
#define sonar40 analog(2)
#define cds_side analog(3)
#define sonar24 analog(4)
#define bend analog(5)
#define cds_box analog(6)
#define SnrTxL                  0x02
#define SnrTxR                  0x01
#define Clear           0xC0
#define bit3            0x08
#define bit4            0x10
#define bit5            0x20
#define sonar_max       24
#define sonar_thrhld    75
#define low_degree 1000
#define center_degree 2650
#define search_delay 2670
#define increment 5
#define pad_motor       0
#define maxtime 200
/************************ End of Defines ******************************/
```

```
/*************************** Prototypes
**********************************/
void main_init(void);
void adjust_speed(int, int);
void motor(int, int);
void straight(void);
void fwd_right(int);
void fwd_left(int);
void bk_right(int);
void bk_left(int);
void bk_straight(int);
void obstacle_avoid(void);
void start_search(void);
void search(void);
void sonar_rt(void);
void sonar_lf(void);
void search(void);
void start_pad(void);
void adjust_pad(void);
void launch(void);
void laun_seq(void);
/*************************End of Prototypes
*****************************/

/*************************** Globals **********************************/
int main_cnt = 0;
int speed = 30;
int rev_speed = -30;
int center = 3000;
int right = 2700;
int left = 3300;
int ir_thrhldlf = 105;
int ir_thrhldrt = 105;
int ir_maxlf = 110;
int ir_maxrt = 110;
int present_speed0 = 0;
int present_speed1 = 0;
int speed_delay = 87;                    /* delay = 174 for loop corresponds to 2 ms*/
int irert = 0x40;
int irelf = 0x80;
int sonarrt_cnt = 0;
int sonarlf_cnt = 0;
int up, count, j, high, found, high_count;/* globals for search start and launch*/
int degree, location, high_degree;
int gotit =0;
int launched = 0;
int movedelay = 30;
/*************************End of globals ******************************/

/*************************  Main   ************************************/
void main(void)
{
  main_init();
  while (irrt < ir_maxrt);         /* Wait until the right ir is set high to turn on*/
  while(1)
  {
	for(main_cnt=0; main_cnt<maxtime; main_cnt++)
	{
	obstacle_avoid();
/*	printf("The analog0 value is %d     The analog1 value is %d\n", irrt, irlf);*/
	}
	if(launched == 0)
	{
		motor(1, 0);
		laun_seq();
		launched = 1;
	}
  }
}
/*************************** End Main  *********************************/

void main_init(void)
/*************************** initialization ****************************
 * Description
 * all initializations
 *
 *
 *Usage:

**********************************************************************
/
{
	int i;
/* Initialize the analog, motors, servo, clock, and serial */
  init_analog();
  init_motortk();
  init_clocktk();
  init_servos();
  init_serial();
  DDRD = 0x38;                    /* set Port D bit 3, 4, and 5 as output bits*/
  CLEAR_BIT(PORTD, bit3);                      /*Clear port d bit 3*/
  poke = 0x00;            /* turn off all 40 kHz signals */
  for(i=0; i<100; i++); /* wait for ir to shut off*/
  ir_thrhldrt = irrt + 5; /* Initialize thresholds values */
  ir_thrhldlf = irlf + 5; /* Assumes position where vehicle is reset*/
  ir_maxrt = irrt + 10;             /* is away from all obstacles*/
  ir_maxlf = irlf + 10;
  poke = 0xC0;            /* turn on right and left ir 40 kHz signals */
  motor(0,0);             /* Initialize motors to 0*/
  motor(1,0);
}
/**************************** initializaion *****************************/

/**************************** obstacle_avoid *****************************
 * Description
 * Steers the vehicle in the appropriate direction depending on the ir
 * sensor and sonar sensor readings.
 *
 *Usage: to control robots direction and speed.

*************************************************************************
/
void obstacle_avoid(void)
{
        int i, k;               /* integer for a counter*/
                sonar_lf();                                     /*check left sonar. returns
counter value*/
                sonar_rt();
            printf("Sonar left is %d and the Sonar right is %d\n", sonarrt_cnt, sonarlf_cnt);
/* check if the left side of the robot is at the minimum distance from an
*  object.  if back up to the left.                                  */
                if(irrt>ir_thrhldrt)
                {
                        bk_right();
                }
                else if(irlf>ir_thrhldlf)
                {
                        bk_left();
                }
                else if(sonarrt_cnt<sonar_max && sonarlf_cnt<sonar_max)
                {
                        bk_left();
                }

                else if(sonarrt_cnt<sonar_max && sonarrt_cnt<sonarlf_cnt)
                {
               bk_right();
                }
```

```
/* check if the left side of the robot is approaching an object.  if so
*  turn to the right.                                              */
                else if(sonarrt_cnt<sonar_thrhld && sonarrt_cnt<sonarlf_cnt)
                {
              fwd_left();
                }

/* check if the right side of the robot is at the minimum distance from an
*  object.  if back up to the right.                               */
                else if(sonarlf_cnt<sonar_max && sonarlf_cnt<sonarrt_cnt)
                {
               bk_left();
                }

/* check if the right side of the robot is approaching an object.  if so
*  turn to the left.                                               */
                else if(sonarlf_cnt<sonar_thrhld && sonarlf_cnt<sonarrt_cnt)
                {
               fwd_right();
                }

/* otherwise go straight at a constant speed                  */
        else
                {
                        straight();
                }
}
/*************************** obstacle_avoid ****************************/


void motor(int index, int percent)
/*************************** motor ************************************
 * Description
 * Sets the speed and direction for the motors.
 *
 *
 *
 *Usage: to controls motor direction

**********************************************************************
/
{
        printf("I'M in the motor function\n");
        if ( index == 0)
        {printf("index = 0\n");
```

```
                /* Set the Direction for motor0*/
                if ( percent < 0)
                {       SET_BIT(PORTD, 0x20);     /*For reverse  SET PD5 to high */
                        motortk(index, percent);        /* Call motor command*/
                }
                else
                {       CLEAR_BIT(PORTD, 0x20);         /* Set PD5 to low */
                        motortk(index, percent);        /* Call motor command*/
                }

        }
        if (index == 1)
        {printf("index = 1\n");
                /* Set the direction of the motor1 */
                if ( percent < 0)
                {       SET_BIT(PORTD, 0x10);                   /* Set PD4 to high*/
                        motortk(index, percent);        /* Call motor command*/
                }
                else
                {       CLEAR_BIT(PORTD, 0x10);         /* Set PD4 to low*/
                        motortk(index, percent);        /* Call motor command*/
                }

        }
}
/*************************** motor **** *****************************/

void straight(void)
/*************************** straight *****************************
 * Description
 * Sets the front servo to straight and motortk to motortk speed.
 *
 *
 *Usage: to control robots direction

**********************************************************************
/
{       printf("Going straight");
        motor(1, speed);
        servo(0, center);
}
/*************************** straight *****************************/

void fwd_right(void)
/*************************** fwd_right *****************************
 * Description
 * Sets the front servo to the right and motortk to motortk speed.
 *
 *
 *Usage: to control robots direction

****************************************************************************
/
{
	printf("Going right");
	motor(1, speed);
	servo(0, right);
}
/**************************** fwd_right *****************************/

void fwd_left(void)
/**************************** fwd_left *****************************
 * Description
 * Sets the front servo to the left and motortk to motortk speed.
 *
 *
 *Usage: to control robots direction

****************************************************************************
/
{
		printf("Going left");
		motor(1, speed);
		servo(0, left);
}
/**************************** fwd_left ****************************/

void bk_right(void)
/**************************** bk_right ****************************
 * Description
 * Sets the front servo to the right in order to turn right backing up
 * and motortk to the backwards direction with half motortk speed.
 *
 *Usage: to control robots direction

****************************************************************************
/
{
		printf("Backing up right");
		motor(1, rev_speed);
		servo(0, right);
}
```

```
/**************************** bk_right *****************************/

void bk_left(void)
/**************************** bk_left *****************************
 * Description
 * Sets the front servo to the left in order to turn left backing up
 * and motortk to the backwards direction with half motortk speed.
 *
 *Usage: to control robots direction

**********************************************************************
/
{
		printf("Backing up left");
		motor(1, rev_speed);
		servo(0, left);
}
/**************************** bk_left *****************************/

void bk_straight(void)
/**************************** bk_straight *****************************
 * Description
 * Sets the front servo to straight and motortk to motortk rev_speed.
 *
 *
 *Usage: to control robots direction

**********************************************************************
/
{
		printf("Backing up straight");
		motor(1, rev_speed);
		servo(0, center);
}
/**************************** bk_straight *****************************/

void sonar_lf(void)
/**************************** sonar_lf *****************************
 * Description
 * pulses left sonar transmitter for 1ms checks the sonar receiver
 * for a pulse
 *
 *Usage: to control robots direction

**********************************************************************
/
```

```
{                           int i;
                            sonarlf_cnt = 0;
                            poke = SnrTxL;
                            for (i = 0; i < 100; i++);
                            poke = Clear;
                            for(i=0; i<100; i++);
                            while((sonar40 > 200) && (sonarlf_cnt < 300))
                            {
                                    sonarlf_cnt++;
                            }
                            if(sonarlf_cnt<=10)
                            {
                                    sonarlf_cnt = 300;
                            }

                            printf("%d \n\n", sonarlf_cnt);
}
/**************************** sonar_lf *****************************/

void sonar_rt(void)
/***************************** sonar_rt *****************************
 * Description
 * pulses right sonar transmitter for 1ms checks the sonar receiver
 * for a pulse
 *
 *Usage: to control robots direction

**********************************************************************
/
 {                          int i;
                            sonarrt_cnt = 0;                /* initialize counter to 0*/
                            poke = SnrTxR;                          /*turn sonar transducer on*/
                            for (i = 0; i < 100; i++);      /* wait for 1 ms*/
                            poke = Clear;                           /*turn sonar transducer off*/
                            for (i = 0; i<100; i++);        /*delay to reduce effects of
transference of signal*/
                            while((sonar40 > 200) && (sonarrt_cnt < 300))
                            {
                                    sonarrt_cnt++;
                            }
                            if(sonarrt_cnt<=10)
                            {
                                    sonarrt_cnt = 300;
                            }
                            printf("%d \n\n", sonarrt_cnt);
}
```

```
/****************************sonar_rt********************************


/****************************start_pad************************
/ Start function moves pad form center point to full right
/ starting postion to begin searching for brightest area.
/*********************************************************/
void start_pad(void)
{
        degree = center_degree;             /* initialize degree to center position*/
        while(degree>low_degree)
        {
                degree = degree - increment;
                servo(1, degree);
      for(j=0; j<search_delay; j++);    /*delay loop to wait 30 ms*/
        }
}
/****************************start_pad*******************************/


/********** Search function  ****************************************
 * Description
 * searches for the highest value of light seen by the cds cell in the 3
 * sided box located on the arm attached to the launch pad
 *
 *Usage: to control the launch pads direction

*************************************************************************
/
void search(void)
{               /* initialize variables*/
          up = 0;
          degree = 1000;
          high_degree = 1000;
          j=0;
          count=0;
          found = 0, high_count = 0;
          location=1000;
          high=255;
      while(gotit == 0)
      {
           if(degree<4400 && found == 0)
           {
              if (cds_side<=high)
              {
                high=cds_side;
                high_degree=degree;
              }
              degree = degree+increment;
              location=degree;
           }
           else if (found == 0)
           {
                found=1;
           }
           if (found == 1 && degree>=high_degree)
           {
                                    degree = degree - increment;
                               }
                               else if (found == 1 &&  degree<high_degree)
                               {
                                    gotit = 1;
                               }
           else
           {
                degree=degree;
           }
           servo(1, degree);
           for(j=0; j<search_delay; j++);        /*delay loop to wait 30 ms*/
      }
}

/********************** search ************************************

/ ******************* adjust_pad ***********************************/
void adjust_pad(void)
{
        int found =0;
        int down = 1;
        int gotit = 0;
        int high = 255;
        int angle = 120;          /* variable used to store angle of highest light value*/
        while(ready == 0)
        {
                if((bend<135) && (down == 1))
                {
                        motor(pad_motor, -100);
                        printf("Motor going down");
                        if ( cds_box < high)
                        {
                                high = cds_box;
                                angle = bend;
                        }
                }
                else if (bend>angle)
                {
                        motor(pad_motor, 100);
                        printf("Motor going up");
                        down = 0;
                }
                else
                {
                        motor(pad_motor, 0);
                        printf("Done !!");
                        found = 1;
                }
        }
}
/*********************************** adjust_pad *********************/

void launch(void)
/*************************** launch ****************************
 * Description
 * sets port d bit 3 to high for 10 ms causing a fuse to ignite and launch
 * the rocket
 *
 *
 *Usage:

**********************************************************************
/
 {
        int i, j;
        for (j=0; j<20; j++)
        {
                for(i=0; i<10000; i++)
                {
                        SET_BIT(PORTD, 0x08);     /*turn port d bit 3 on for 7 s*/
                }
        }
        CLEAR_BIT(PORTD, 0x08);                   /* turn port d bit 3 off*/
        printf("Launch successfull");
 }
/****************************** launch *******************************/

void laun_seq(void)
/******************************laun_seq********************************
***
```

```
 * Description
 * executes appropriate sequence of functions to launch rocket.
 *
 *
 *
 *Usage:

***********************************************************************
/
{
        start_pad();
      search();
      adjust_pad();
      launch( );
}
/****************************** laun_seq ******************************/
```