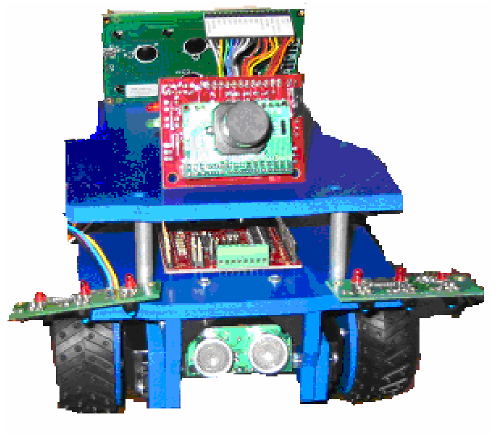


T.R.O.N.

Transportional Regulation Obedient Newbie

Dima Haddad
08/01/05

TAs:
William Dubel
Steven Pickles

Instructors:
A.A. Arroyo
E. M. Schwartz

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

TABLE OF CONTENTS

ABSTRACT

The T.R.O.N. (Transportation Regulation Obedient Newbie) robot is a car simulation robot. It will react accordingly to a traffic signal on a road way. The robot will be able to stay between to solid lines (much like a lane on a roadway), avoid collision with other vehicles or objects on the road, and discern between the red, yellow, and green phase of a traffic signal and behave as a real world driver to those phases. The main sensors that I am planning to use would be sonar for proximity detection and object avoidance, CMU cam for vision and color detection, IR for the line avoidance, and bump sensors just incase of an object rear ending the robot. For the object avoidance the robot will only stop in its path and not turn to avoid objects, because on a roadway swerving into another lane to avoid impact is at times more dangerous to the driver and other road users than the collision would have been. If an object is detected with in six inches the robot will stop and honk its horn until the object is removed. The IR sensors will be used for the line avoidance. There will be two on either side of the robot (in the front) so that it stays between the two white lines. The CMU cam will be used to discern which light (green, yellow, or red) is on so that the robot will be able to have the right reaction to the traffic light.

EXECUTIVE SUMMARY

The purpose of this project is to create an autonomous vehicle that can simulate a drive along an arterial roadway and adhere to traffic signals. This robot is T.R.O.N. (Transportation Regulation Obedient Newbie). It reacts accordingly to a traffic signal on a road way. The robot is able to stay between to solid white lines (much like a lane on a roadway), avoid collision with other vehicles or objects on the road, and discern between the red, yellow, and green phase of a traffic signal and behave as a real world driver to those phases. The main sensors that I am planning to use are IR sensors, sonar sensor, bump sensor, and a CMUcam. There are two IR sensors one on each side of the front of the robot and they detect the lines and avoid them. If one of the IRs detects a line it turn in the opposite direction and than reverts to it path. The sonar sensor is used for proximity detection and object avoidance. For the object avoidance the robot will only stop in its path and not turn to avoid objects, because on a roadway swerving into another lane to avoid impact is at times more dangerous to the driver and other road users than the collision would have been. If an object is detected with in six inches the robot will stop and honk its horn until the object is removed. Another form of object detection is the one bump sensor on the back of the robot which when pushed in the robot will also stop for a time of two and half seconds. The last sensor is the CMUcam for vision and color detection. The CMUcam takes a snapshot of the traffic light when it is triggered and relays the information to the brain which reacts with a certain motion.

As far as the priority, collision avoidance is first on the list (which includes first the sonar and then the bump sensors), then the line following. The camera function is only called when it is triggered.

This robot also has an environment that is similar to a roadway. The background is black and there is a loop of white lines one on the inside of the other to make a track. There are two traffic lights one on either side of the straight away.

To conclude the robot performs all of it functions with little problems except for maybe certain lighting conditions. However the darker the area the better it seems to work. The global implication of the project is to someday be able to remove the human factor from the roadway.

INTRODUCTION

In Transportation Engineering, one of the first lessons includes the defining of the main three aspects of design of a roadway. These three aspects are vehicle characteristics (length, horsepower, height, etc…), environment (roadway type, number of lanes, weather, etc…), and driver characteristics. Out of these three the driver characteristic is the hardest of the three to account for in the design of a roadway facility, it includes the drivers state of mind and his/her abilities and disabilities. This is the problem that can be solved by the addition of intelligent systems to a roadway facility as well as vehicles. This project will focus on the latter of these two, in hopes of one day to mostly (or if possible completely) removing the human factor from the design equation, and creating a much safer driving environment for all.

There has been large scale and much more intricate projects that have successfully created an autonomous vehicle that can maneuver through inner city arterials, drive on freeways, and even maneuver through desert terrain over a certain distance (to learn more Google “autonomous vehicle”). However, there is a reason that this robot’s name ends with the word “Newbie”, for this project will only replicate a small part of previous and ongoing projects for autonomous vehicles. It will concentrate on the aspect of recognizing and reacting to a traffic light while staying in its designated lane of traffic.

The T.R.O.N. project will entail several things of which the main parts will be the building of the robot and of its test environment. The robot itself will have a brain, a body, and sensors to interact with the world around it. The environment is a miniature roadway model with a lane and traffic signals. It has a black background representing the asphalt and white lines for the lane edges. The traffic signal is overhead of the vehicle as in the real world and will have the red, yellow, and green phase lights. In the following paragraphs all these aspects will be discussed in detail.

INTEGRATED SYSTEM

The whole system of sensors and electronics will be run by a Mavric – IIB board that has an ATMEGA 128 chip on board. The power supply for all the components is one battery pack which approximately delivers 11 Volts (eight rechargeable "AA" batteries).

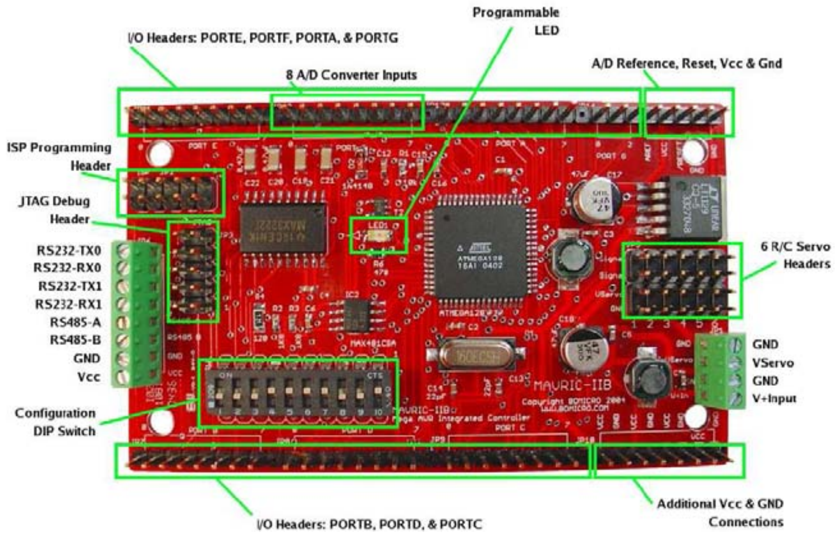
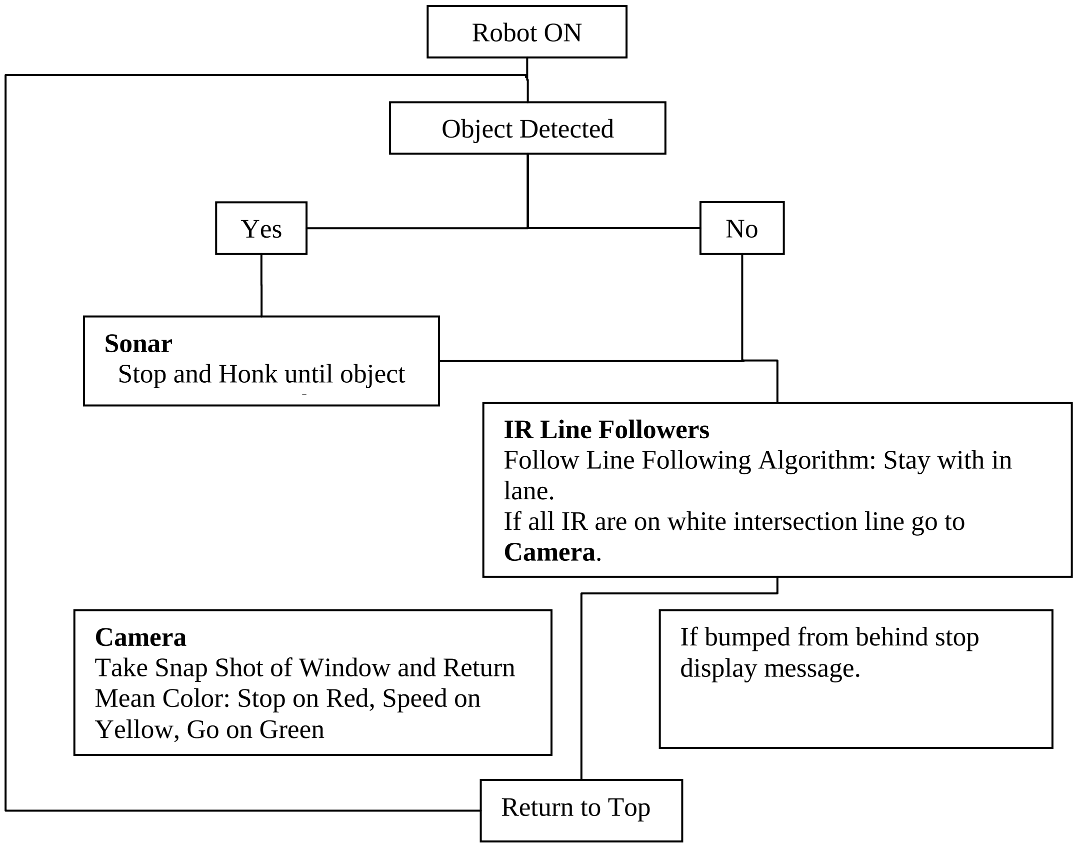


Figure 1: Mavric – IIB (BDMicro)

The components that are to be controlled by the Mavric – IIB board are the LCD, two line following IR sensors, one SRF04 sonar sensor, one CMUcam, a bump switch, LCD, and a buzzer. The way the system is organized is in the following diagram:



Robot ON
Object Detected
Yes
No
Sonar
Stop and Honk until object
IR Line Followers
Follow Line Following Algorithm: Stay with in lane.
If all IR are on white intersection line go to Camera.
Camera
Take Snap Shot of Window and Return Mean Color: Stop on Red, Speed on Yellow, Go on Green
If bumped from behind stop display message.
Return to Top

PLATFORM

Platform Design

The platform will slightly resemble a vehicle. The platform consists of two main "T" shaped PVC board which is 6mm thick. The dimensions for these "T" boards are 6.75" in length by 5.125" in width. Figure 1 consists of AutoCAD drawings and renderings of the boards (Budget Robotics):

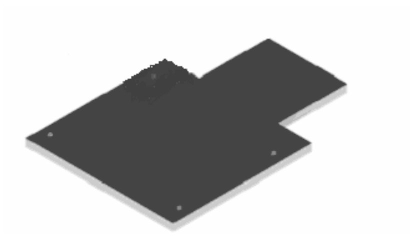
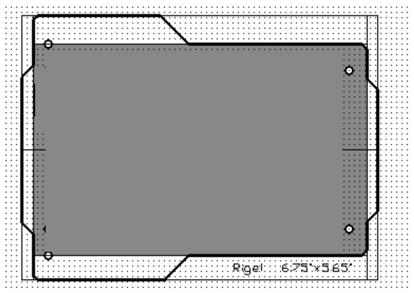
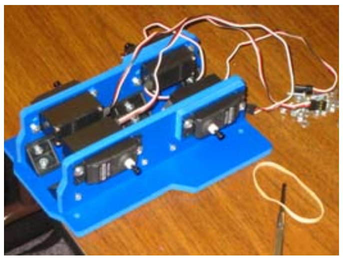
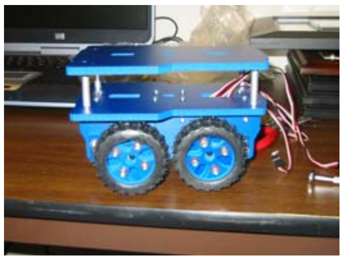



Figure 2: Robot Base Visual (Rigel, Budget Robotics)

Platform Implementation

The T.R.ON. Robot will look like a miniature monster truck. The "T" base described would be the ideal platform due to the 60 sq inches of space that allows all the sensors and other equipment to be mounted on boards. Also since this is a vehicle robot the tires must be durable. The wheels that are mounted to the robot have the ability to traverse various terrains (such as, carpet, tile, grass, concrete, asphalt, and dirt.) The measurements for the complete body of the robot with the wheels attached are L: 6.75", W: 6", and H: 4.25". Secondly since there are to main "T" section boards, the robot has two levels, which will be separated by rises approximately 1 ¼" apart. The following figures are pictures of the assembly process for the robot body:





Figures 3, 4, & 5: Periodic Pictures of robot platform assembly.

Lessons Learned: The body of the robot is extremely important, because the design of it must accommodate the additional parts that are to be attached to it in the future. Next time (which will also give more time) I would take time planning the whole robot design and build the body my self. Since I bought this body I had to compromise with the placement of my sensors.

ACTUATION

The robot platform will run on four wheels, each with its own individual servo. A view of the wheels and servos are shown below:

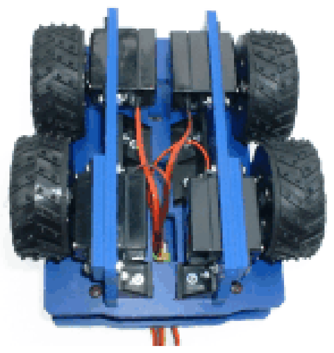


Figure 6: Bottom view of platform, with wheels and servos (Budget Robotics).

The wheel and servo sets have the following specifications given by the retailer (Budget Robotics):

- Tire diameter: 65mm (2 1/2"); tread 7/8" wide
- Tire material: Medium-hardness treaded rubber (with "studs" for traction)
- Hub: Custom machined from PVC plastic
- Futaba-spline wheel hub, to match Futaba R/C servos

The R/C servos have been modified for continuous (360^{o}) rotation and need any where between 4.8 to 6 volts of power to operate. In this project a regulated five volts will be used to power the servos as well as all other components on the robot. Another important point is that since the servos on opposite sides of the robot are mirror images of each other if the same pulse width is sent to both one side will go in one direction while the other side will go in the opposite direction, this should be taken into account when programming the servos for movement.

Other than the wheels the T.R.O.N. robot will not have other actuation components.

Lessons Learned: Next time I would use DC motors just for the speed aspect of the project.

SENSORS

There are several sensors that are need for the robot to accurately mimic a car on the roadway. The following is a list of sensors to be used along with some miscellaneous parts:

- Two IR Sensors
- One Sonar Sensor
- Four Bump Sensors
- CMU Cam
- Miscellaneous: buzzer and various leds

IR Sensors

There are two IR line tracking sensors to be used on the robot body. Each IR sensors will be placed on either side of the front of the robot about ½" off the ground. They will also be placed so the outer edge of the sensor board is further out than the robot body.

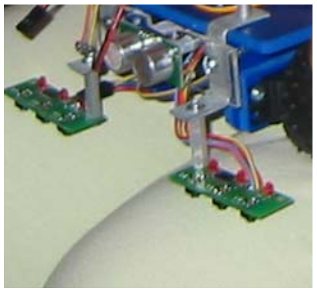


Figure 7: IR Sensor Mounting

The IR sensors will detect the difference between light and dark backgrounds. They go high for white surfaces and low for dark. The surfaces do not have to be black and white. The IR sensors also work in various light conditions. They have been tested in window light (daytime), window light plus intense light fixtures, and at night time with low intensity light bulbs that are not placed in the robots line of sight. The IR sensor has worked in all these situations. They also work for distance of about 2 mm away from the surface to a little more than ½" away from a surface; they will not however work if they are touching the surface. The following are some schematics of the IR sensor being used:

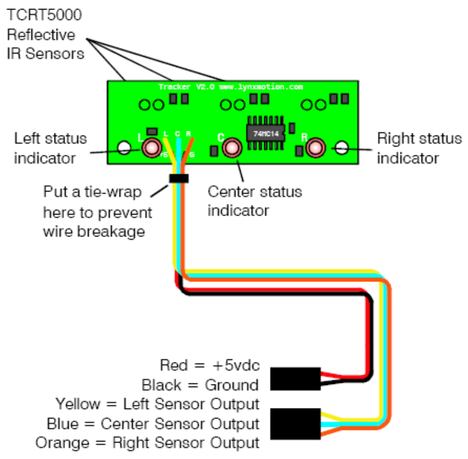


Figure 8: Diagram of IR sensor (Lynxmotion, Inc)

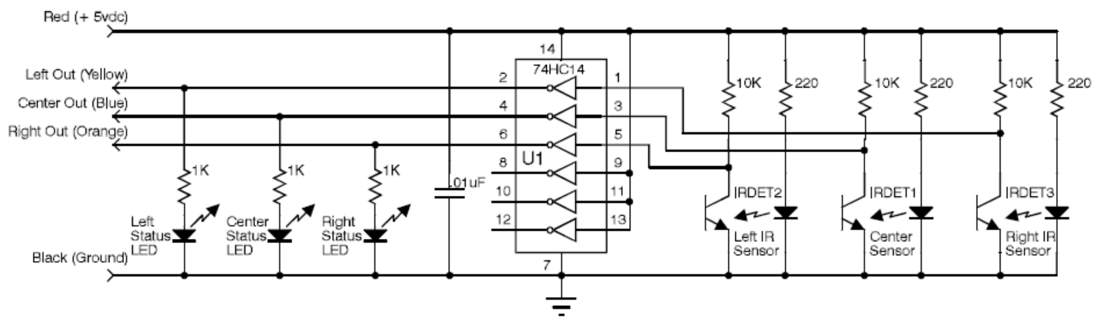


Figure 9: Circuit Schematic Diagram (Lynxmotion, Inc)

The IR sensor on the T.R.O.N. robot is used to avoid white lines on either side of the robot that represent a lane of traffic. When the right sensor detects the white line it causes the robot to turn left and vice versa for the left sensor. Secondly, since there are three IRs on each sensor, the closer the line is to the inner most sensor on one side, the more adjustment to the right or left the robot makes. After the adjustment is done the robot reverts back to a forward motion.

Lessons Learned: The after the first demo until right before the demo day the line following was not to smooth and seemed to over estimate the adjustment of the turn, to fix this problem the reversion back towards forward motion was done at a faster speed than the turn correction which made the line following much smoother in the end.

Sonar Sensor

The sonar sensor will be used for proximity detection so the robot does not collide with any other objects on the roadway. The robot only needs one sonar sensor due to the fact that the robot stops before an object and does not swerve into another lane or oncoming traffic. Once the sonar detects an object the robot stops and will not move until the object is removed.

The way the sonar works is it sends out a ping and waits for an echo to return and then measures the distance as the function of the time. The sonar used on the robot is a SRF04, the following list are the specifications of this sensor (Lynxmotion, Inc.):

- Sensor type = Reflective Ultrasonic
- Frequency = 40KHz
- Ultrasonic sender = N1076
- Ultrasonic receiver = N1081
- I/O required = Two digital lines, 1 output, 1 input
- Minimum range = Approximately 3cm
- Maximum range = Approximately 3m
- Sensitivity = Detects a 3cm diameter stick at > 2m
- Input trigger = 10uS Min. TTL level pulse
- Echo Pulse = Positive TTL level signal, width proportional to range
- Input voltage = 5vdc regulated
- Current requirements = 30mA Typ 50mA Max
- PC board size = ~.75" x 1.75"

The sonar sensor on the T.R.O.N. robot is set to determine distance in inches. The minimum range for the sonar during testing was 1" and the maximum range around 9 ft (108"), the max angle with which it detected an object was approximately 25º.

The following charts are of the timing and beam pattern of the SRF04.

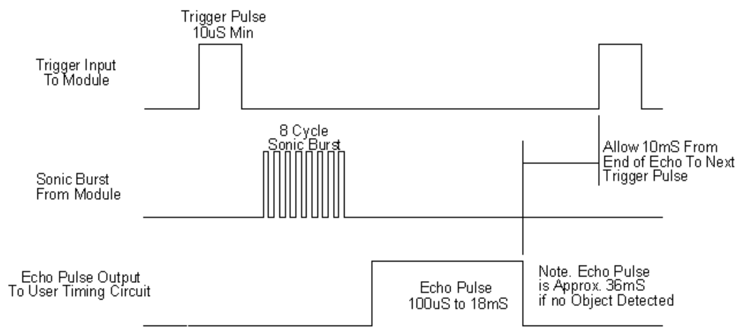


Figure 10: Timing Chart for Sonar SRF04

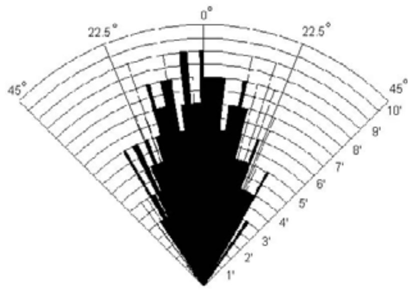


Figure 11: Beam Pattern for Sonar Sensor

Another collision avoidance tool will be the bump sensor which is used in the case an object collides with T.R.O.N from the back. If this occurs T.R.O.N will stop and scream at the object/vehicle "Whiplash!, Whiplash!" and then continue on its path after 2.5 seconds.

CMU Camera

The CMU cam will be used for vision, so the robot can see the traffic light as well as which phase it is in (i.e. red, yellow, or green). Once the image is processed it will react accordingly, which would be stop for red, go for green, and double current speed for yellow. Below is a picture of the CMU cam from Seattle Robotics:

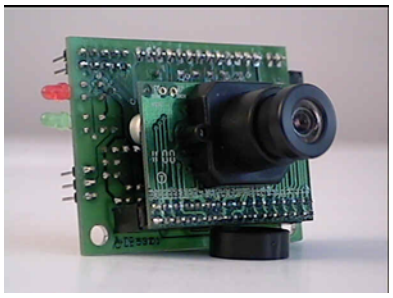


Figure 12: CMU Cam Board (Seattle Robotics)

The way the camera works: (Rowe, et al., 2002)

> Upon completion of the frame, it divides these accumulated values by the total number of pixels returning the mean color. It also returns an approximation of the absolute deviation from the mean of each color. This can be used like a variance measure to quantify the spread of the colors about the mean. When used in conjunction with other features such as windowing, described below, the color statistics can be used as a building block for a motion detection algorithm or for determining the color of an object at a specific location in the field of view.

Since the robot is only going to be using the CMU cam to detect three colors the “GM\R” command (or get mean color value) will be used. The mean values produced by the camera are between the range of 16 to 240.

In testing the CMU cam lighting was the most difficult variable to account for. To minimize the effect of the lighting conditions the camera’s auto white balance and gain are on for the calibrating phase and then are turned off to better notice the shifts in color. The camera does calibrate to the traffic light in front of it to a set window of coordinates (1, 60) to (60, 83) for six seconds. After that the camera is triggered by the an intersection line on the roadway, in which it takes a snapshot of the traffic light and gives back a packet of R G B colors. While testing on the traffic light several times it was noticed that the camera detected the red light with the average packet numbers at R 215 G 190 and the yellow light at R 176 G 140. The Blue color never changed significantly and the default was the green light detected.

Miscellaneous

The miscellaneous section has parts in it that will allow the robot to look and be more car-like, such as having a horn and front and rear lights.

BEHAVIORS

This robot will have four distinct behaviors which are

- Object Detection: If any object colloids with the robot from the back and triggers the bump switch the robot will stop.

- Line Tracking: The robot will track and stay within two solid white lines, that represent a lane of an arterial.
- Collision Avoidance: If any object is within six inches of the front of the robot the robot will stop.
- Vision: The robot will calibrate to the traffic signal in front of it and will react when the different lights turn on. Stop for Red, Speed Up for Yellow, and Go for Green.

CONCLUSION

The T.R.O.N. robot up to date as accomplished the objects that have been set out in this paper. The robot drives within its designated lane of traffic, it stops for objects at the most six inches away from the front of the body, it stops for rear end collision that it is involved in, uses the camera to detect which light is on and react accordingly to it, and all this is done with feedback to the LCD screen telling the viewer what is going on.

This project was challenging but fun. What exceeded expectation was the line following, which never started with a very high expectation originally. The object of most problems was the CMUcam and the vision behavior, lighting plays too much of a role in this arena, this is the one thing that would most definitely need more work on this particular robot. Vision enhancement is a must if it is to be put to practical use.

References

BDMICRO. Mavric – IIB Board User's Manual.
http://www.bdmicro.com/images/mavric-iib.pdf

Budget Robotic. Rigel Construction and Operation.
http://www.oricomtech.com/rigel/rgl-info.htm

Lynxmotion, Inc. Users Manual TRA-01 Version 5.0. &
http://www.lynxmotion.com/Product.aspx?productID=57&CategoryID=8 &
http://www.lynxmotion.com/Product.aspx?productID=59&CategoryID=8

Rowe, et al. A Low Cost Embedded Color Vision System. Jan. 19, 2002.
http://www-2.cs.cmu.edu/~cmucam/Publications/iros-2002.pdf

Seattle Robotic. CMUcam Users Manual.
http://www.seattlerobotics.com/New%20CMUcam%20manual%20.doc

Acknowledgements

This section is here to acknowledge the enormous help that I had from Fernando Hernandez. Although I know the basic knowledge behind the logic of programming, I do not know a language in its entirety. I do not want to take credit for something I did not do and probably could not have done with out a couple of years of programming under my belt, so I would like to say that most of the code that is in the appendix I did get from him such as the LCD function, the motor functions, the sonar function, and the CMUcam get mean function. I wrote down in logical plain english how I wanted the behaviors to work and he translated in to "C" code for me. Although most of my function code was borrowed all of it was explained to me almost line by line so I do understand what it does.

APPENDIX

Code for Robot

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>

volatile uint16_t ms_count;
volatile uint16_t us_count;

// 20x4 LCD Screen *******************************************************************
// LCD DATA
        LCD Control
// Port Pin        0        1        2        3        4        5        6        7
        Port Pin        0        1        2
// LCD Pin                  DB0      DB1 DB2 DB3 DB4 DB5 DB6 DB7                  LCD Pin
        RS       RW       EN
#define LCD_DATA                  PORTE
#define LCD_CTRL                  PORTF
#define LCD_DATA_DDRX    DDRE
#define LCD_CTRL_DDRX    DDRF

// SRF04 ***************************************************************************
// Port Pin        0        1        2        3        4        5        6        7
// LCD Pin                  OUT TRG
//        --#1---
#define SRF_PORT          PORTC
#define SRF_DDRX          DDRC
#define SRF_PINX          PINC

// IR Line Trackers *******************************************************************
// PORT Pin    0  1  2  3  4  5  6  7
// IR Pin    RHT CNT LFT    LFT CNT RHT
//                          ----LEFT----          ----Right---
#define IR_Data                  PORTA
#define IR_DDRX                  DDRA
#define IR_PINX                  PINA

// Servo motors *********************************************************************
#define L_MOTOR          OCR1A
#define R_MOTOR          OCR1B
#define M_CENTER         18500

// delay for specified number of milliseconds
void ms_sleep(uint16_t ms)
{
  TCNT0  = 0;
  ms_count = 0;
  while (ms_count != ms*21);
}

// delay for specified number of microseconds * 48
void us_sleep(uint16_t us)
{
  TCNT0  = 0;
  us_count = 0;
  while (us_count != us);
}

// millisecond counter interrupt vector
SIGNAL(SIG_OUTPUT_COMPARE0)
{
  ms_count++;
  us_count++;
}

// initialize timer 0 to generate an interrupt every 48usec

void init_timer(void)
{
  TIFR  |= _BV(OCIE0);
  TCCR0  = _BV(WGM01)|_BV(CS02)|_BV(CS00);// CTC, prescale = 128
  TCNT0  = 0;
  TIMSK |= _BV(OCIE0);         // enable output compare interrupt
  OCR0   = 6;                          // match in 48usec
}

// ******************************************************************************************************************************
// Initializes the display ***************************************************************************************************
// ******************************************************************************************************************************
void lcd_initialize(void)
{
        LCD_CTRL_DDRX = 0x07;     // enable lower 3 bits of control port
        LCD_DATA_DDRX = 0xFF;     // enable entire data port

        LCD_DATA = 0x38;          // set for 8 bit mode, 1/16 duty cycle
        LCD_CTRL = 0x04;          // enable high
        ms_sleep(1);                      // wait
        LCD_CTRL = 0x00;          // enable low
        ms_sleep(1);                      // wait

        LCD_DATA = 0x0C;          // turn display on, cursor off, and set no blink
        LCD_CTRL = 0x04;
        ms_sleep(1);
        LCD_CTRL = 0x00;
        ms_sleep(1);

        LCD_DATA = 0x06;          // set auto increment (to write forwards)
        LCD_CTRL = 0x04;
        ms_sleep(1);
        LCD_CTRL = 0x00;
        ms_sleep(1);
}
```

```
// Turns the cursor on - cursor("on") and off -  cursor ("off")
void lcd_cursor(char command[])
{
        if(command[1]==116 || command[1]==110){
                LCD_DATA = 0x0E;
                LCD_CTRL = 0x04;
                ms_sleep(1);
                LCD_CTRL = 0x00;
                ms_sleep(1);}
        else{
                LCD_DATA = 0x0C;
                LCD_CTRL = 0x04;
                ms_sleep(1);
                LCD_CTRL = 0x00;
                ms_sleep(1);}
}

// Goes to the beginning of the line specified by its argument (valid ranges = 1 to 4)
void lcd_goto_line(int line)
{
        switch(line)
        {
                case(1): {LCD_DATA = 0x80; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00; ms_sleep(1);
break;}
                case(2): {LCD_DATA = 0xC0; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00; ms_sleep(1);
break;}
                case(3): {LCD_DATA = 0x94; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00; ms_sleep(1);
break;}
                case(4): {LCD_DATA = 0xD4; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00; ms_sleep(1);
break;}
        }
}

// Goes to a line specified by the argument (valid ranges = 1 to 4) and
// then a position on that line (valid ranges = 1 to 20)
void lcd_goto_pos(int line, int pos)
{
        switch(line)
        {
                case(1): {LCD_DATA = 0x80+pos-1; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00;
ms_sleep(1); break;}
                case(2): {LCD_DATA = 0xC0+pos-1; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00;
ms_sleep(1); break;}
                case(3): {LCD_DATA = 0x94+pos-1; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00;
ms_sleep(1); break;}
                case(4): {LCD_DATA = 0xD4+pos-1; LCD_CTRL = 0x04; ms_sleep(1); LCD_CTRL = 0x00;
ms_sleep(1); break;}
        }
}

// Clears all text from the screen
void lcd_clear_display(void)
{
        LCD_CTRL = 0x04;
        LCD_DATA = 0x01;
        ms_sleep(1);
        LCD_CTRL = 0x00;
        ms_sleep(1);
}

// Outputs a single ASCII character to the screen
void lcd_put_char(char c)
{
        LCD_CTRL = 0x05;
        LCD_DATA = c;
        ms_sleep(1);
        LCD_CTRL = 1;
        ms_sleep(1);
}

// Outputs a string message (character array) to the screen
void lcd_put_string(char message[])
{
        int i;
        for(i=0;i<21;i++)
        {
                if(message[i]=='\0') break;
                else lcd_put_char(message[i]);
        }
}

// Flashes a text message on a specified line, every specified
// number of milliseconds, a specified number of times.
void lcd_flash_message(int line, int delay, int num_of_times, char message[] )
{
        int flag = 0;
        num_of_times = num_of_times * 2;
        for(;num_of_times>0;num_of_times--)
        {
                if(flag==0)
                        {ms_sleep(delay); lcd_goto_line(3); lcd_put_string(message); flag=1;}
                else
                        {ms_sleep(delay); lcd_goto_line(3); lcd_put_string("                    "); flag=0;}
        }
}

// Displays the intro text
void lcd_show_intro(void)
{
        int delay = 100;
        lcd_clear_display();
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string(".");
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string("N.");
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string(".N.");
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string("O.N.");
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string(".O.N.");
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string("R.O.N.");
        ms_sleep(delay); lcd_goto_line(2); lcd_put_string(".R.O.N.");
```

```
		ms_sleep(delay); lcd_goto_line(2); lcd_put_string("T.R.O.N.");
		ms_sleep(delay); lcd_goto_line(2); lcd_put_string(" T.R.O.N.");
		ms_sleep(delay); lcd_goto_line(2); lcd_put_string("  T.R.O.N.");
		ms_sleep(delay); lcd_goto_line(2); lcd_put_string("    T.R.O.N.");
		ms_sleep(delay); lcd_goto_line(2); lcd_put_string("     T.R.O.N.");
		ms_sleep(delay); lcd_goto_line(2); lcd_put_string("      T.R.O.N.      ");

		ms_sleep(delay); lcd_flash_message(3,250,3," Boot Simulation... ");
}

// ******************************************************************************************************************
// Function to get distance data from an SRF04 **************************************************************************
// ******************************************************************************************************************
int SRF04(int times)
{
		int i,counter = 0;

		for (i=0; i<times ;i++)
		{
				ms_sleep(50);		// wait 50msec between pulses for echo to settle

				// Trigger a ping on coresponding SRF04 and wait a few usec

				SRF_DDRX = 0x02;			// set output pin
				SRF_PORT = 0x02;			// trigger
				us_sleep(8);					// wait a few usec
				SRF_PORT = 0x00;			// end trigger

				// wait a few usec for echo circuit initialization
				us_sleep(8);

				while(1)
				{
						if(SRF_PINX & 0x01){		// check if echo has been received
								counter++;				// increment counter
								us_sleep(4);}			// wait a few usec
						else break;					// break if echo was received
				}
		}
		return ((counter/times)*1.8);			// return average (calibrated for inches)
}

// ***************************************************************************************************************
// Motor Drive Functions ******************************************************************************************
// ***************************************************************************************************************
void motors_initialize(void)
{

		// count up to 20,000 at a rate 8x slower than the 16MHz clock
		// yielding 10ms to count up, and 10ms to count down = 20 ms period
		ICR1 = 20000;

		// Enables OC1 and OC3 on all channels (A, B, and C)
		// OC bits will set on upcount and clear on downcount, and the counters
```

```
	// take their TOP value from the ICRx register. (waveform generation mode #8)
	//	7		6		|	5		4		|
	3		2		|	1		0
	//	Compare Mode	|	Compare Mode	|	Compare Mode	|
	Waveform Generation
	//	Channel A		|	Channel B		|	Channel C
	|	Bits 1 and 0
	TCCR1A = 0xFC;

	// Set prescalter to 8x slower than chip clock, set other half of
	// waveform generation mode
	//	7		6		|	5		|	4
	3		|	2		1		0
	//	ICNC	ICES	|	Reserved|	Wave Generation	|	Clock Select Bits
	//					|			|	Bits 3 and 2
	|
	TCCR1B = 0x12;

	// Set timer to zero (16 bit timers)
	TCNT1 = 0x0000;

	// Sets ports to outputs.(OC1A ,B, C)
	// Pin outputs for A, B, C are B5, B6, B7, respectively
	DDRB = 0xE0;

	// M_CENTER both servos
	L_MOTOR = M_CENTER;
	R_MOTOR = M_CENTER;
}

// Stops the motors
void motors_stop(void)
{
	int l_incrementor, r_incrementor;

	// if motors are already centered, just return
	if(L_MOTOR == M_CENTER && R_MOTOR == M_CENTER) return;

	// if the left motor is currently going backwards, set L incrementor to positive, else negative
	if(L_MOTOR < M_CENTER) l_incrementor = 1;
	else l_incrementor = -1;

	// if the right motor is currently going backwards, set L incrementor to positive, else negative
	if(R_MOTOR < M_CENTER) r_incrementor = 1;
	else r_incrementor = -1;

	// change their speed until desired speed is reached
	while(L_MOTOR != M_CENTER)
	{
		L_MOTOR = L_MOTOR + l_incrementor;
		R_MOTOR = R_MOTOR + r_incrementor;
		ms_sleep(3);
	}
}
```

```
//=================================================================
// Move in a specified direction, at a specified speed
void motors_move(char direction[], int speed)
{
	int i;

	// Forwards, decrement left motor, increment right motor
	if(direction[0]==70 || direction[0]==102)
	{
		if(L_MOTOR == M_CENTER - speed && R_MOTOR == M_CENTER + speed) return;
		else motors_stop();

		// change their speed until desired speed is reached
		for(i=1;i<=speed;i++)
		{
			L_MOTOR = L_MOTOR - 1;
			R_MOTOR = R_MOTOR + 1;
			ms_sleep(3);
		}
	}

	// Backwards, increment left motor, decrement right motor
	if(direction[0]==66 || direction[0]==98)
	{
		if(L_MOTOR == M_CENTER + speed && R_MOTOR == M_CENTER - speed) return;
		else motors_stop();

		// change their speed until desired speed is reached
		for(i=1;i<=speed;i++)
		{
			L_MOTOR = L_MOTOR + 1;
			R_MOTOR = R_MOTOR - 1;
			ms_sleep(3);
		}
	}

	// Left turn, increment both motors
	if(direction[0]==76 || direction[0]==108)
	{
		if(L_MOTOR == M_CENTER + speed && R_MOTOR == M_CENTER + speed) return;
		else motors_stop();

		// change their speed until desired speed is reached
		for(i=1;i<=speed;i++)
		{
			L_MOTOR = L_MOTOR + 1;
			R_MOTOR = R_MOTOR + 1;
			ms_sleep(3);
		}
	}

	// Right turn, decrement both motors
	if(direction[0]==82 || direction[0]==114)
	{
```

```
                if(L_MOTOR == M_CENTER - speed && R_MOTOR == M_CENTER - speed) return;
                else motors_stop();

                // change their speed until desired speed is reached
                for(i=1;i<=speed;i++)
                {
                        L_MOTOR = L_MOTOR - 1;
                        R_MOTOR = R_MOTOR - 1;
                        ms_sleep(3);
                }
        }
}

// ===========================================================================
// CMUCam Functions ==========================================================
// ===========================================================================

volatile int MAX_MSG_SIZE = 30;
volatile unsigned char CMUResponseBuffer[15];

// initialize UART1 to 38.4k baud rate
void UART1_init(void)
{

        UBRR1H = 0x00;
        UBRR1L = 0x33;
        UCSR1A |= 0x02;
        UCSR1C = 0x06;
        UCSR1B = 0x18;
}

// transmit a message (char array) over UART1
void USART1_Transmit(char data[MAX_MSG_SIZE])
{
        int t = 0;
        while ((t < (MAX_MSG_SIZE + 1)) & (data[t] != 0x00))
        {
                // wait for empty transmit buffer
                while ((UCSR1A & _BV(UDRE1)) == 0);
                UDR1 = data[t];
                t++;
        }
}

// wait for data to be received and then returns it
// NOTE: this is a BLOCKING receive!
unsigned char USART1_Receive( void )
{
        while ( !(UCSR1A & (1<<RXC1)) );
        return UDR1;
}

void CMU_init(void)
{
	USART1_Transmit("RS\r");	// reset
	ms_sleep(20);
	USART1_Transmit("PM 1\r");	// poll mode
	ms_sleep(20);
	USART1_Transmit("RM 3\r");	// raw output
	ms_sleep(20);
	USART1_Transmit("MM 1\r");	// middle mass on
	ms_sleep(20);
	USART1_Transmit("SW\r");	// full screen
	ms_sleep(20);

}

// queries the CMU cam to get the mean values of R, G, B
// stores them in the global "CMUResponseBuffer"
void CMU_GetMean(void)
{
	int i = 0;
	char tempChar;

	USART1_Transmit("GM\r");

	// initial 255 framing byte is read in, discarded
	tempChar = USART1_Receive();

	// this command returns a "type S" packet, 7 bytes long
	// we read in those 7 bytes
	for(i=0;i<7;i++)
	{
		CMUResponseBuffer[i] = USART1_Receive();
	}

	// last 255 framing byte is read in, discarded
	while(tempChar != ':')
	{
		tempChar = USART1_Receive();
	}

	CMUResponseBuffer[i] = '\0';
}


// tells the CMUCam to track a color
void CMU_TrackColor(int Rmin, int Rmax, int Gmin, int Gmax, int Bmin, int Bmax)
{
	int i = 0;
	char tempChar, tempMessage[30];

	sprintf(tempMessage,"TC %i %i %i %i %i %i\r",Rmin, Rmax, Gmin, Gmax, Bmin, Bmax);

	USART1_Transmit(tempMessage);

	// initial 255 framing byte is read in, discarded
	tempChar = USART1_Receive();
	// this command returns a "type M" packet, 9 bytes long
	// we read in those 9 bytes
	for(i=0;i<9;i++)
	{
		CMUResponseBuffer[i] = USART1_Receive();
	}

	// last 255 framing byte is read in, discarded
	while(tempChar != ':')
	{
		tempChar = USART1_Receive();
	}

	CMUResponseBuffer[i] = '\0';
}

// used to convert the raw data returned by the CMUCam to an integer
int binary2int(unsigned char binary_num)
{
	int result = 0;
	if(binary_num & 1) result +=0;
	if(binary_num & 2) result +=2;
	if(binary_num & 3) result +=4;
	if(binary_num & 8) result +=8;
	if(binary_num & 16) result +=16;
	if(binary_num & 32) result +=32;
	if(binary_num & 64) result +=64;
	if(binary_num & 128) result +=128;
	return result;
}
```

Main

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

```
int main(void)
{
	char response[10];
	char message[30];
	int distance = 0;

	int counter=0;

	init_timer();			// initialize timer
	sei();					// enable interrupts

	//Initialize Intro :)
	lcd_initialize();
	UART1_init();
	ms_sleep(100);
	CMU_init();
	motors_initialize();
	//lcd_show_intro();
	lcd_clear_display();


	ms_sleep(500);
	USART1_Transmit("SW 1 60 60 83\r");
	ms_sleep(500);
	lcd_goto_line(1);lcd_put_string("Calibrating....  ");
	USART1_Transmit("CR 18 44 19 33\r");
	ms_sleep(3000);
	ms_sleep(3000);
	USART1_Transmit("CR 18 40 19 32\r");
	ms_sleep(500);

	int R, G, B;
	lcd_clear_display();

	while(1)
	{
		motors_move("F",30);

		// Line detection and display code*********************************************************************

		IR_DDRX = 0x00;

		lcd_goto_pos(1,1);
		// if interior left is on white
		if(IR_PINX & 0x01){
			//lcd_put_string("Interior Left ");
			motors_stop();
			while(IR_PINX & 0x01)
			{
			if((IR_PINX & 0x01) && (IR_PINX & 0x20) && (IR_PINX & 0x02) && (IR_PINX & 0x40) &&
				(IR_PINX & 0x04) && (IR_PINX & 0x80)){goto detect_light;}
				motors_move("R",15);
			}continue;
		}
```

```
// if interior right is on white
if(IR_PINX & 0x20){
        //lcd_put_string("Interior Right");
        motors_stop();
        while(IR_PINX & 0x20)
        {
        if((IR_PINX & 0x01) && (IR_PINX & 0x20) && (IR_PINX & 0x02) && (IR_PINX & 0x40) &&
                (IR_PINX & 0x04) && (IR_PINX & 0x80)){goto detect_light;}
        motors_move("L",15);
        }continue;
}


// if middle left is on white
lcd_goto_line(2);
if(IR_PINX & 0x02){
        //lcd_put_string("Middle Left   ");

        counter=0;

        while(IR_PINX & 0x02)
        {
                if((IR_PINX & 0x01) && (IR_PINX & 0x20) && (IR_PINX & 0x02) && (IR_PINX &
                        0x40) && (IR_PINX & 0x04) && (IR_PINX & 0x80))
                        {
                                while(counter>0)
                                {
                                L_MOTOR = L_MOTOR + 9;
                                R_MOTOR = R_MOTOR + 9;
                                counter--;
                                ms_sleep(10);
                                }
                                goto detect_light;
                        }
                L_MOTOR = L_MOTOR - 9;
                R_MOTOR = R_MOTOR - 9;
                counter++;
                ms_sleep(200);
        }

        while(counter>0)
        {
        L_MOTOR = L_MOTOR + 9;
        R_MOTOR = R_MOTOR + 9;
        counter--;
        ms_sleep(75);
        }continue;
}
// if middle right is on white
if(IR_PINX & 0x40){
        //lcd_put_string("Middle Right  ");
        counter=0;

        while(IR_PINX & 0x40)
```

```
		{
			if((IR_PINX & 0x01) && (IR_PINX & 0x20) && (IR_PINX & 0x02) && (IR_PINX &
				0x40) && (IR_PINX & 0x04) && (IR_PINX & 0x80))
				{
					while(counter>0)
					{
					L_MOTOR = L_MOTOR - 9;
					R_MOTOR = R_MOTOR - 9;
					counter--;
					ms_sleep(10);
					}
					goto detect_light;
				}
				L_MOTOR = L_MOTOR + 9;
				R_MOTOR = R_MOTOR + 9;
				counter++;
				ms_sleep(200);
		}

		while(counter>0)
		{
		L_MOTOR = L_MOTOR - 9;
		R_MOTOR = R_MOTOR - 9;
		counter--;
		ms_sleep(75);
		}continue;
	}

	// if outer left is on white
	lcd_goto_line(3);
	if(IR_PINX & 0x04){
		//lcd_put_string("Outer Left   ");

		counter=0;

		while(IR_PINX & 0x04)
		{
			if((IR_PINX & 0x01) && (IR_PINX & 0x20) && (IR_PINX & 0x02) && (IR_PINX &
				0x40) && (IR_PINX & 0x04) && (IR_PINX & 0x80))
				{
					while(counter>0)
					{
					L_MOTOR = L_MOTOR + 4;
					R_MOTOR = R_MOTOR + 4;
					counter--;
					ms_sleep(10);
					}
					goto detect_light;
				}
		L_MOTOR = L_MOTOR - 4;
		R_MOTOR = R_MOTOR - 4;
		counter++;
		ms_sleep(200);
		}
		while(counter>0)
		{
		L_MOTOR = L_MOTOR + 4;
		R_MOTOR = R_MOTOR + 4;
		counter--;
		ms_sleep(75);
		}continue;
	}
	// if outer right is on white
	if(IR_PINX & 0x80){
		//lcd_put_string("Outer Right  ");

		counter=0;

		while(IR_PINX & 0x80)
		{
			if((IR_PINX & 0x01) && (IR_PINX & 0x20) && (IR_PINX & 0x02) && (IR_PINX &
				0x40) && (IR_PINX & 0x04) && (IR_PINX & 0x80))
				{
					while(counter>0)
					{
					L_MOTOR = L_MOTOR - 4;
					R_MOTOR = R_MOTOR - 4;
					counter--;
					ms_sleep(10);
					}
					goto detect_light;
				}
			L_MOTOR = L_MOTOR + 4;
			R_MOTOR = R_MOTOR + 4;
			counter++;
			ms_sleep(200);
		}

		while(counter>0)
		{
		L_MOTOR = L_MOTOR - 4;
		R_MOTOR = R_MOTOR - 4;
		counter--;
		ms_sleep(75);
		}continue;
	}

	// Sonar Detection, Bump, and display code************************************************
	// Get and display data for the SRF04

	DDRD = 0x01;
	while(SRF04(1)<6)
	{
		lcd_clear_display(); lcd_put_string("Obstacle detected...");
		motors_stop();
		ms_sleep(3000);
		if(SRF04(1)<6)
		{
			lcd_goto_line(2); lcd_put_string("MOVE JACKASS!");
			PORTD = 0x01; ms_sleep(250); PORTD = 0x00; ms_sleep(250);
			PORTD = 0x01; ms_sleep(75); PORTD = 0x00; ms_sleep(80);
			PORTD = 0x01; ms_sleep(75); PORTD = 0x00; ms_sleep(80);
			PORTD = 0x01; ms_sleep(75); PORTD = 0x00; ms_sleep(80);
			PORTD = 0x01; ms_sleep(100); PORTD = 0x00; ms_sleep(250);
			PORTD = 0x01; ms_sleep(500); PORTD = 0x00; ms_sleep(250);
			PORTD = 0x01; ms_sleep(500);
			while(SRF04(1)<6);
		}
	}
	PORTD = 0x00;


	DDRD = 0x01;
	if(PIND & 0x80)
	{
		lcd_clear_display();
	}
	else
	{
		motors_stop();
		PORTD = 0x00;
		lcd_goto_line(1); lcd_put_string("AHHH!!! OMFGWTF!!");
		lcd_goto_line(2); lcd_put_string("  WHIPLASH!!");
		lcd_goto_line(3); lcd_put_string(" I'm going to call");
		lcd_goto_line(4); lcd_put_string("    my lawyer!");
		ms_sleep(2500);
	}

	continue;

	detect_light:

	// Light detection code
================================================================================
	CMU_GetMean();

	lcd_goto_line(2);
	R = binary2int(CMUResponseBuffer[1]);
	sprintf(message,"%i",R);
	lcd_put_string("R "); lcd_put_string(message);lcd_put_string("  ");

	G = binary2int(CMUResponseBuffer[2]);
	lcd_goto_line(3);
	sprintf(message,"%i",G);
	 lcd_put_string(" G "); lcd_put_string(message);lcd_put_string("  ");

	B = binary2int(CMUResponseBuffer[3]);
	lcd_goto_line(4);
	sprintf(message,"%i",B);
	lcd_put_string(" B "); lcd_put_string(message);lcd_put_string("  ");
```

```
				lcd_goto_line(1);

				if(R > 190  && G > 180)
				{
					lcd_put_string("Red detected!      ");
					motors_stop();
					ms_sleep(3000);
					continue;
				}
				else if(R > 155 && G < 179)
				{
					lcd_put_string("Yellow detected!   ");
					//motors_stop();
					motors_move("F",50);
					ms_sleep(1500);
					continue;
				}
				else
				{
					lcd_put_string("Green detected!    ");
					//motors_stop();
					motors_move("F",30);
					ms_sleep(2000);
					continue;
				}




			}
			return 0;


		}
```