University of Florida
Department of Electrical and Computer Engineering
EEL 5666 – Spring 2006
Intelligent Machines Design Laboratory

Instructor: : Dr A. Arroyo, Dr. E. M. Schwartz
TAs: Adam Barnett, Sara Keen

~~{Wi-Tesla}~~
(Wifi-Power Transfer and Jamming)
Final Report

April 25, 2006

Ahmad EL Kouche "Ed"

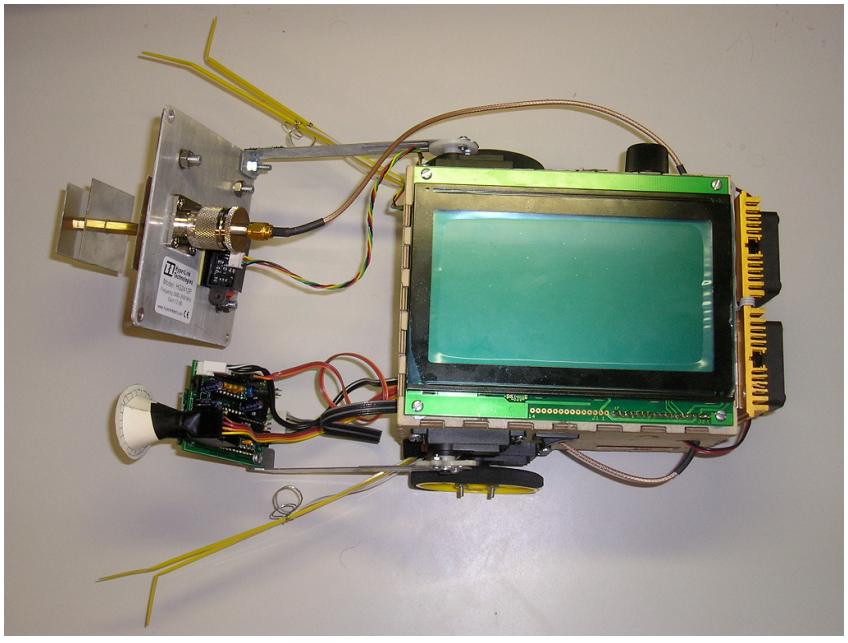


Table of Contents

Abstract

The purpose of this document is to describe Wi-Tesla's design procedure and behavior from platform construction , components, actuation, and system integration. The document will flow from physical component layer and end at the application software layer. Wi-Tesla will present wireless power transfer and ability to jam wifi signals.

Introduction

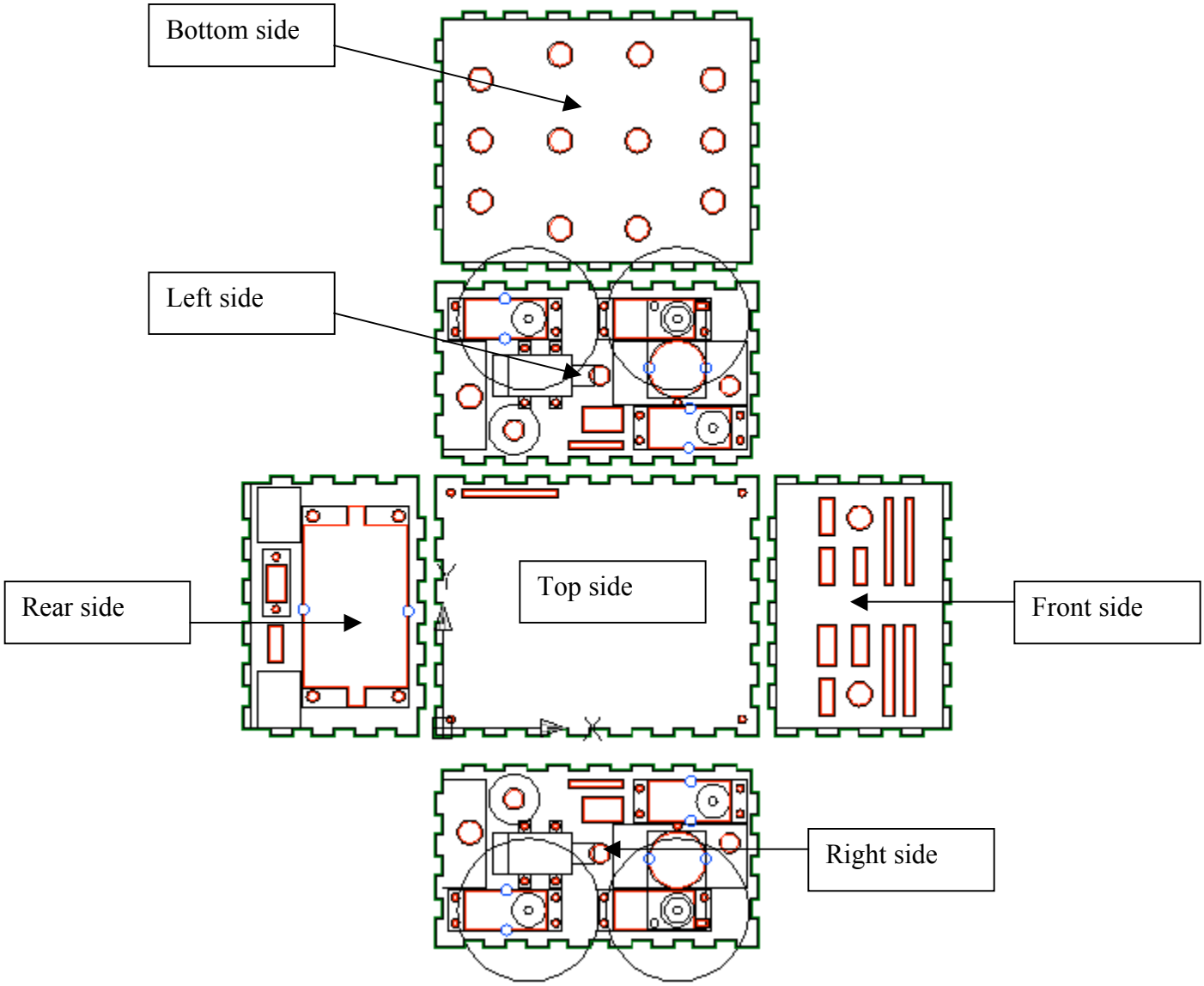
Wi-Tesla is a robot that moves around a given location avoiding obstacles via ultrasonic sound waves, and has collision detection via front whiskers. Wi-Tesla main function is to find a low battery sensor and transfer power to it via radio frequency (RF) energy. The special sensor providing wireless power transfer service is divided into two parts. The first part is the radio frequency generator. The second part is the RF to DC sensor. When Wi-Tesla detects a low battery device in the room via a color tracking camera, the RF energy is radiated toward the sensor, and DC energy is displayed on a multimeter showing power transfer. Since the RF energy radiated is a high power energy that could possibly be harmful to human beings after being exposed for long periods of time, a pyroelectric sensor is used to detect human presence and avoid human radiation.

Robot Physical Platform

The platform was carefully planned and designed in Autocad 2004. Using a digital caliper, all measurements were done based on real component measurements. The general shape of Wi-Tesla is a 3D rectangular shape.

Six faces Description:

Top surface side: the top face holds the graphical LCD
Left surface side: the left side holds the left servo and left arm servo as well as the microcontroller power switch. In addition, a future expansion option is available for the addition of a left-rear wheel.
Right surface side: the right side holds the right servo and right arm servo as well as the RF power switch. In addition, a future expansion option is available for the addition of a right-rear wheel.
Front surface side: this side has many slots for future use, however, only some are used to pass cables into the body.
Rear surface side: this side holds the high power amplifier , two heat-sinks, two fans, temperature sensor, RS232 serial port, and in circuit serial programmer (ICSP).
Bottom surface side: this side contains three casters on the rear side.



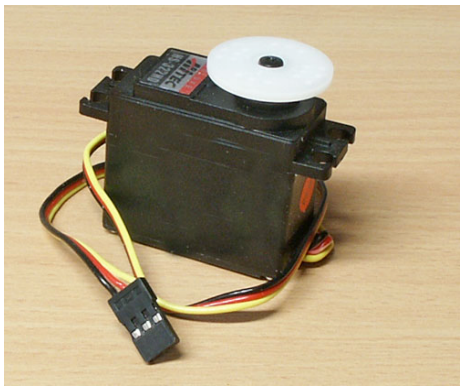
System Components and Behaviors

Wi-Tesla is composed of the following components that are necessary to achieve an autonomous power transfer and signal jamming robot:

Servos

Wi-tesla has four servos in total. Two of them are continuous rotating servos (internally hacked) to serve as rotating wheels. The wheels are responsible to move the robot around. The other two servos, which are not hacked, are used as left arm and right arm.

The servos are HiTec HS-322HD Deluxe
Specification:
Cored Metal Brush Motor
Torque at 4.8V 42 oz/in (3.0 kg/cm)
Torque at 6.0V 51 oz/in (3.7 kg/cm)
Speed at 4.8V 0.19sec/60 degrees at no load
Speed at 6.0V 0.15sec/60 degrees at no load
Dimensions 1.57" x 0.79"x 1.44"
(40 x 20 x 36.5mm)
Weight 1.52oz (43g)



Wheels Behaviors

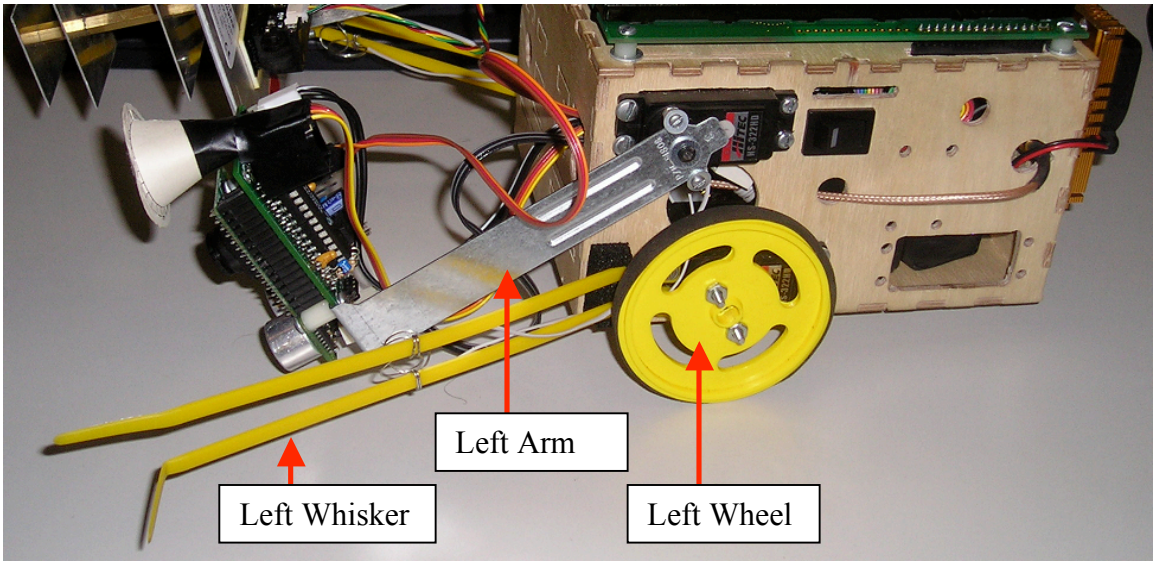
There are basically seven behaviors Wi-Tesla's wheels are capable of doing, and are defined as C functions:

1) Void Left_Turn (int speed): This function will cause Wi-Tesla to take a left turn at a given speed argument (speed0, speed1,speed2,speed3,speed4,speed5). These speeds were noticed to be audibly and visually different than each other, such that speed0 is neutral, also known as "brakes" and speed 5 is the fastest saturation speed accomplished. There is no return value from this function.
   Example: Left_Turn (speed5); //continuous left turn at 5th speed
2) Void Right_Turn (int speed): This has the same behavior as above, however, as indicated it will make a right turn instead.
   Example: Right_Turn (speed5); //continuous right turn at 5th speed
3) Void Turn_ON (int servo_num): This function will turn on the servo specified in the argument value. Values are specified as constants: Right_Servo or Left_Servo.
   Example: Turn_On (Right_Servo); // Right servo on with last motion memory
   Example: Turn_On (Left_Servo); // Left servo on with last motion memory
4) Void Turn_Off (int servo_num): This function will turn off the servo specified in the argument value. Values are specified as constants: Right_Servo or Left_Servo.
   Example: Turn_Off (Right_Servo); // Right servo off (no power consumed)
   Example: Turn_Off (Left_Servo); // Left servo off (no power consumed)
5) Void Forward (int speed, long time): This function moves Wi-Tesla forward at the given speed and for the specified amount of time in m-sec.
   Example: Forward (speed5, 1000);//move forward for one second
6) Void Backward (int speed, long time): This function moves Wi-Tesla backward at the given speed and for the specified amount of time in m-sec.
   Example: Backward (speed5, 1000);//move backward for one second.
7) Move (int Direction, int Servo_Num, int Speed): this function is a general movement function that is optimized for code efficiency and speed. This function will move the specified servo in the given direction and a given speed.
   Example: Move(Go_Forward, Right_Servo, Speed5)// move right servo forward
   Example: Move(Go_Backward, Left_Servo, Speed5)// move left servo backward

Arms Behaviors

Wi-Tesla has two arms located on the left and the right side of the robot. Each arm carries different components and sensors and has a different role of intelligence. The left arm carries the CMUcam, SRF04 UltraSonic range detector, and the PyroElectric sensor. The right arm carries only the Yagi antenna due to its heavy weight. There are basically three behaviors Wi-Tesla's arms are capable of doing, and are defined as C functions:

1) void turn_arm (int arm, int angle): This function will turn the given arm specified in the first argument to the angle specified in the second argument.
   Example: Turn_Arm(Right_Arm, 180)//this will extend the right arm horizontally at a 180 degrees
   Example: Turn_Arm(Left_Arm, 210)//this will extend the left arm 210 degrees forward as seen in the picture below
2) Void Turn_ON (int servo_num): This function will turn on the arm specified in the argument value. Values are specified as constants: Right_arm or Left_arm.
   Example: Turn_On (Right_Arm); // Right arm on with last angle in memory
   Example: Turn_On (Left_Arm); // Left arm on with last angle in memory
3) Void Turn_Off (int servo_num): This function will turn off the servo specified in the argument value. Values are specified as constants: Right_Servo or Left_Servo.
   Example: Turn_Off (Right_Arm); // Right arm off (no power consumed)
   Example: Turn_Off (Left_Arm); // Left arm off (no power consumed)



Whiskers

Wi-Tesla has what looks like whiskers. These whiskers are used to detect obstacle collision from left and right side of the robot. The whiskers are homemade from tie-straps. Two tie straps are necessary to form each of the left and the right whisker. When Wi-Tesla hits on obstacle from one side, one of the tie straps will get contact to ground through the grounded metal located on the second tie strap, thus operating as a switch like mechanism. This homemade collision detection switch is easy, works great, and is cost effective.

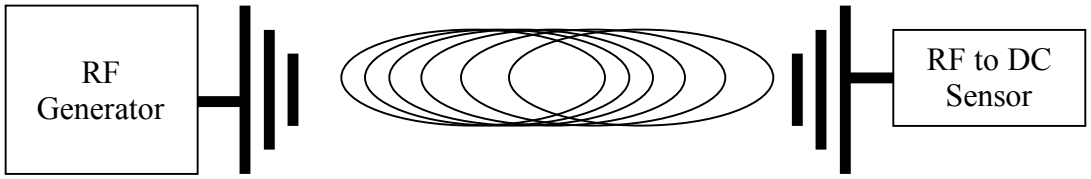
The whiskers have one common behavior:

1) Void Check_Bumps(Void): This function will check each of the bumps for possible collision, and would make an opposite turn to the location of the collision detected.

Wireless Generator and Receiver

Wi-Tesla main function is to find a low battery sensor and transfer power to it via radio frequency (RF) energy. The special sensor providing wireless power transfer service is divided into two parts. The first part is the radio frequency generator. The second part is the RF to DC sensor. When Wi-Tesla detects a low battery device in the room via a color tracking camera, the RF energy is radiated toward the sensor, and DC energy is displayed on a multimeter showing power transfer.

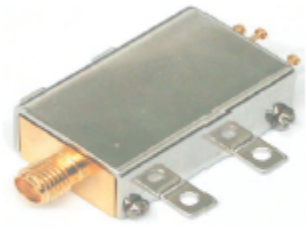
The wireless energy transfer system is composed of two entities. The first entity is the radio generator. The second entity is the RF to DC sensor.



Radio Generator

The radio generator is a system that provides high radio frequency power via a high frequency oscillator, high power amplifier, and high directivity antenna. The oscillator is a high frequency generator ZX95-2650 from www.minicircuits.com , cost is 44$ each. Some electrical features of this device are the following:

DC operation: Vcc at 12 volts with maximum current draw of 25mA
Frequency Tuning: 2165 MHz – 2650 MHz
Power Output: 5dBm

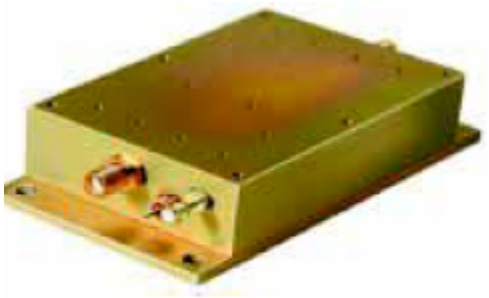


CASE STYLE: GB956
PRICE: $43.95 ea. QTY (1-9)

More features of this oscillator can be found at this link:
http://www.minicircuits.com/ZX95-2650.pdf

The high power amplifier is the ZRL-3500 from www.minicircuits.com, cost is 140$ each. The amplifier is fed with the signal generated form the oscillator. The signal is amplified with a gain of about 20dB and then fed to a high directivity gain antenna. Some electrical features of the amplifier are the following:

Frequency: 700 MHz to 3500MHz
Gain: 26 dB (700-1600MHz), 21 dB (1600 – 2600MHz), 16 dB (2600-3500MHz)
DC operation: 12volts with maximum current draw of 575mA



CASE STYLE: FJ893

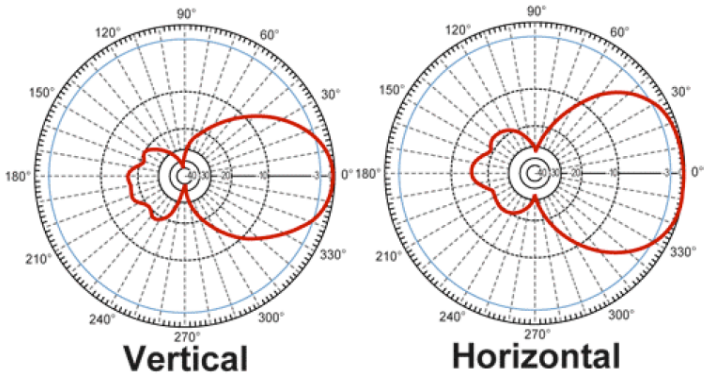
Model	Price
ZRL-3500	$139.95 ea.

The amplified signal is then fed through a high directivity antenna HG2412P from www.hyperlinktechnologies.com, cost 33$ each. The antenna has about 12 dB of gain in the zero degree vertical and horizontal direction.
Some electrical features of the amplifier are the following:

Gain: 12 dB
Polarization: vertical (34 degree) and horizontal (65 degree)

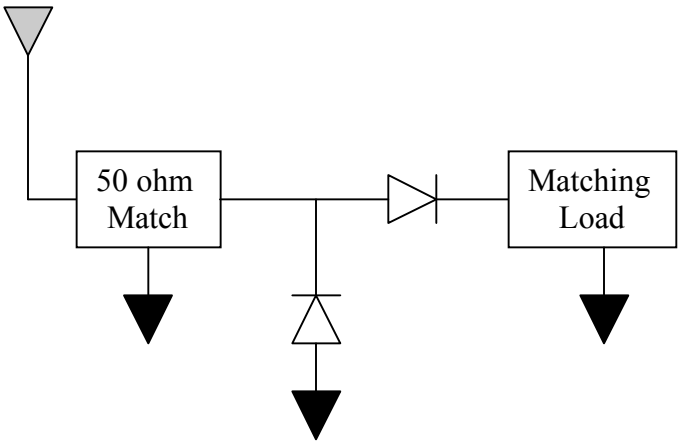
Max input power: 200W
VSWR: 1.5:1

Radiation Pattern



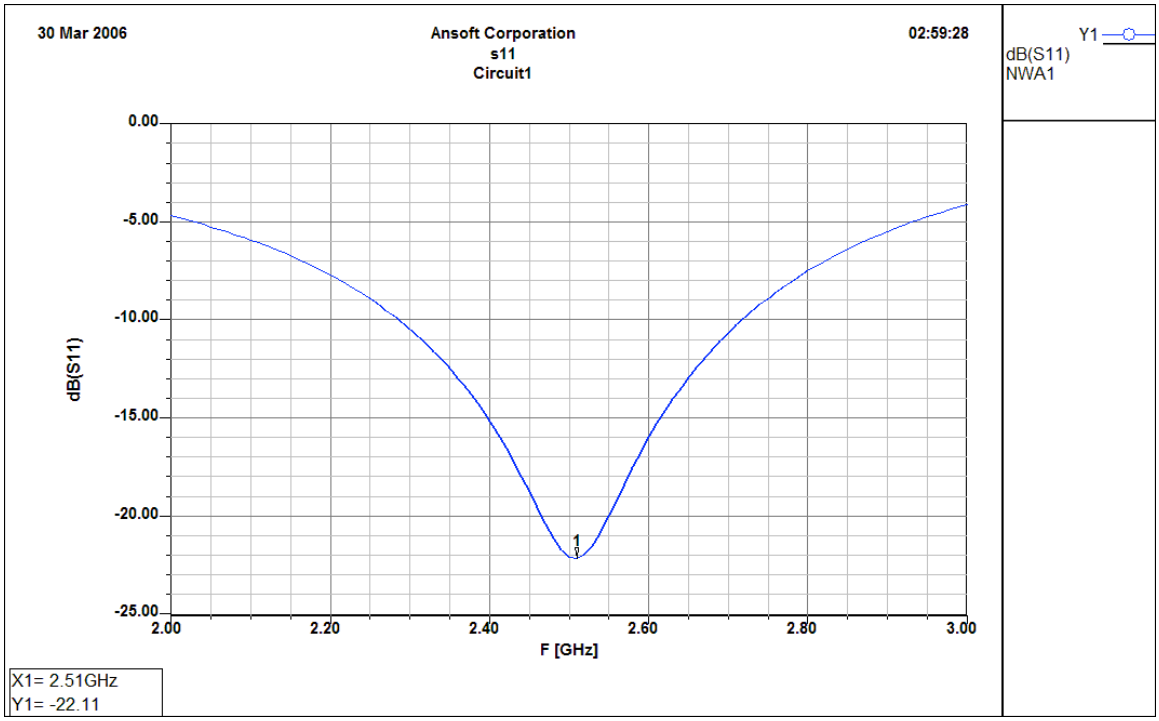
RF to DC Sensor

This sensor requires high radio frequency design expertise. The RF to DC sensor was simulated on Ansoft Designer software. Basically, the sensor is composed of RF detection schottcky diode, which is a special zero biased diode. The input of the sensor has to be matched to the desired operating frequency. Therefore, input matching network is required for maximum power transfer. The detection diodes are placed in a voltage doubling setup for higher voltage output. The sensor schematic is shown below.

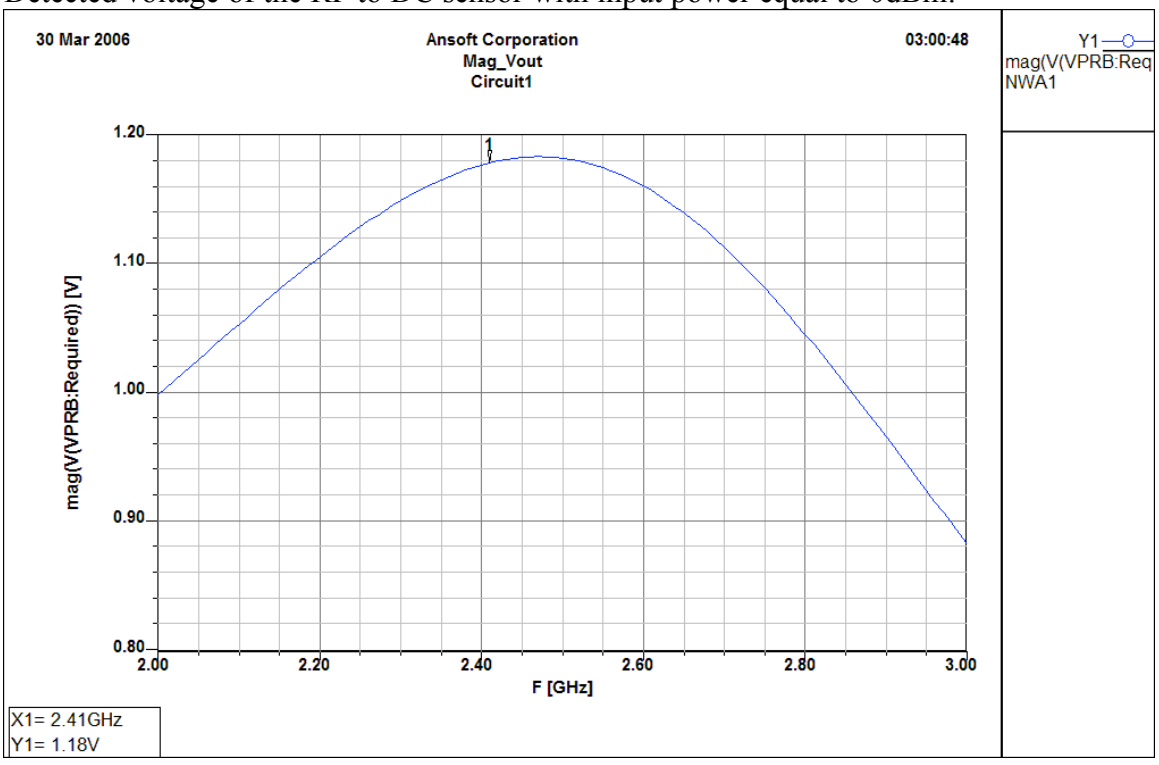


The antenna receives the signal from the RF generator, passes through the RF detection diodes, and then the energy is transformed from radio to DC at the matched load.

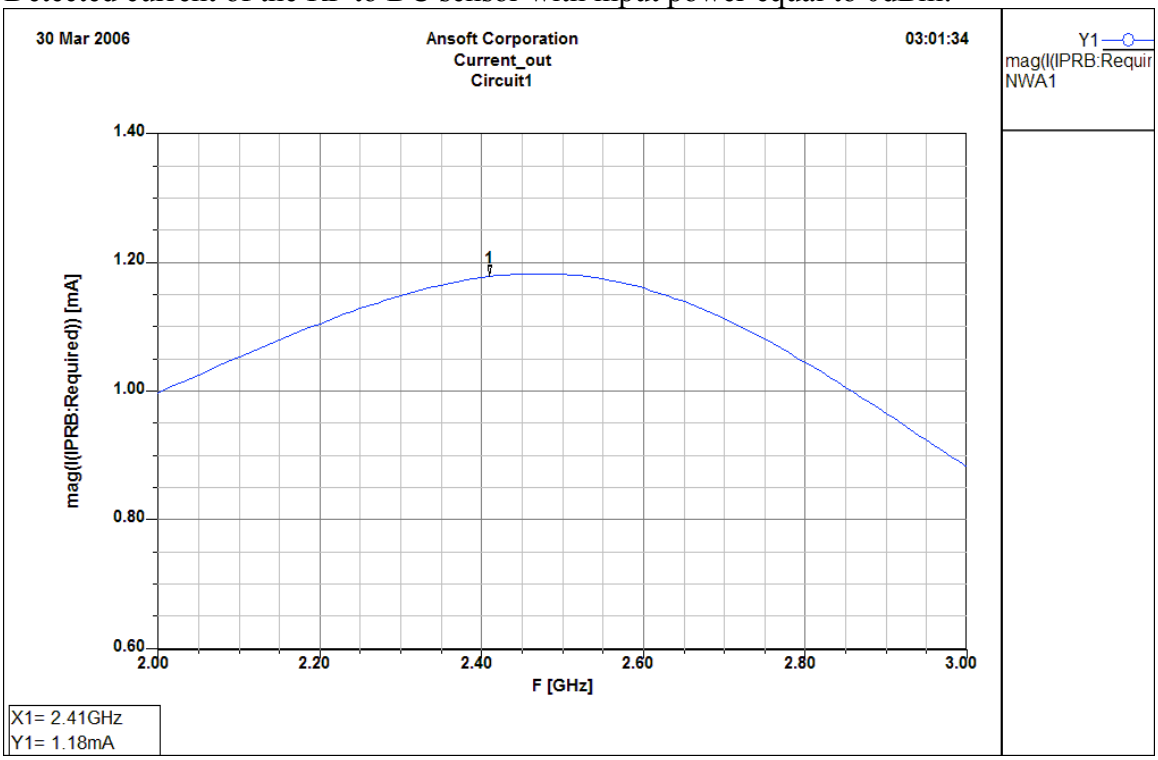
Some electrical features of the amplifier are the following:
Return Loss S11 of the RF to DC sensor:



Detected voltage of the RF to DC sensor with input power equal to 0dBm:



Detected current of the RF to DC sensor with input power equal to 0dBm:



Graphical and Text LCD

At the top side of Wi-Tesla is a graphical LCD that is capable of displaying up to 16 lines of text by 32 characters across, graphical capability display of 256x128 pixels. The LCD has a SED1330 onboard controller. This LCD is available from allelectronics for about $18.
The LCD can be a little complicated to work with at first, but once the controller data sheet is understood, the functions become understandable.

LCD Functions:

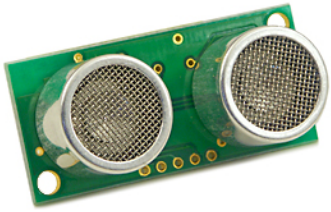
```
void gl_init();                 // initializes LCD display
void gl_grfclr();               // clears graphical LCD pixels
void gl_grfhome();              // sets cursor addresse to graphical
void gl_strobe();               // shifts data into LCD
void gl_txtclr();               // clears all text on screen
void gl_setaddr   (int gl_cur, long gl_addr);     //sets address of cursor
void send_data    (int gl_byte);                  // sends data, then Strobes data in
void send_command (int gl_cmd);                   // sends command data to LCD
void g_printf     (char string[],long Line_num);  //prints a text line at the line number
void g_clear_line (long Line_num);                // clears the specified line
```

UltraSonic Range Detector (SRF04)

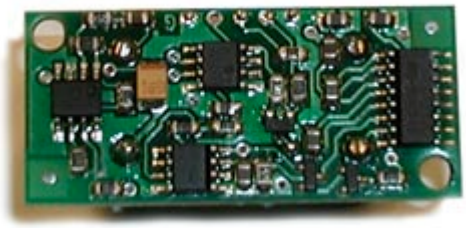
The ultrasonic range detector is located on the left arm of Wi-Tesla. The arm is angled at 180 degrees (horizontal position) to obtain a straight distance measurement. Distance is obtained as a floating point integer in cm.

Ultrasonic Functions:

1) float get_distance (); // this function takes one distance measurement and will return a floating point distance in cm
   Example: dist = get_distance();// dist has nearest distance reading in cm
2) float get_average_distance (int num_average); // this function takes a number of distance measurements, average obtained distances, and will return a floating point average distance in cm.
   Example: avg_dist = get_average_distance(10);//avg_dist has 10 distance average readings



Front View



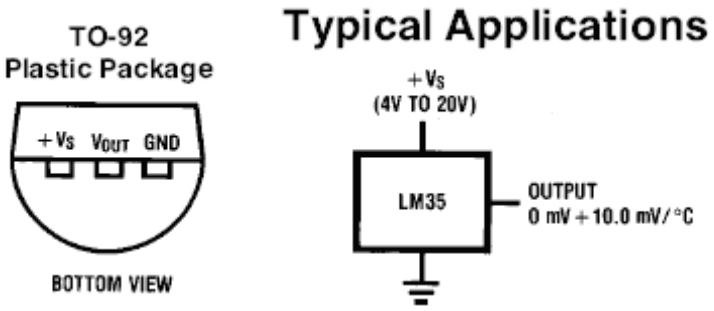
Back View

Temperature Sensor (LM35CAZ)

Wi-Tesla requires a temperature sensor to monitor the temperature of the high power amplifier onboard. The need for the temperature sensor is essential because the amplifier can easily damage itself due to continuous power consumption (0.8 Amps). Also, linearity of Gain is a function of Temperature.

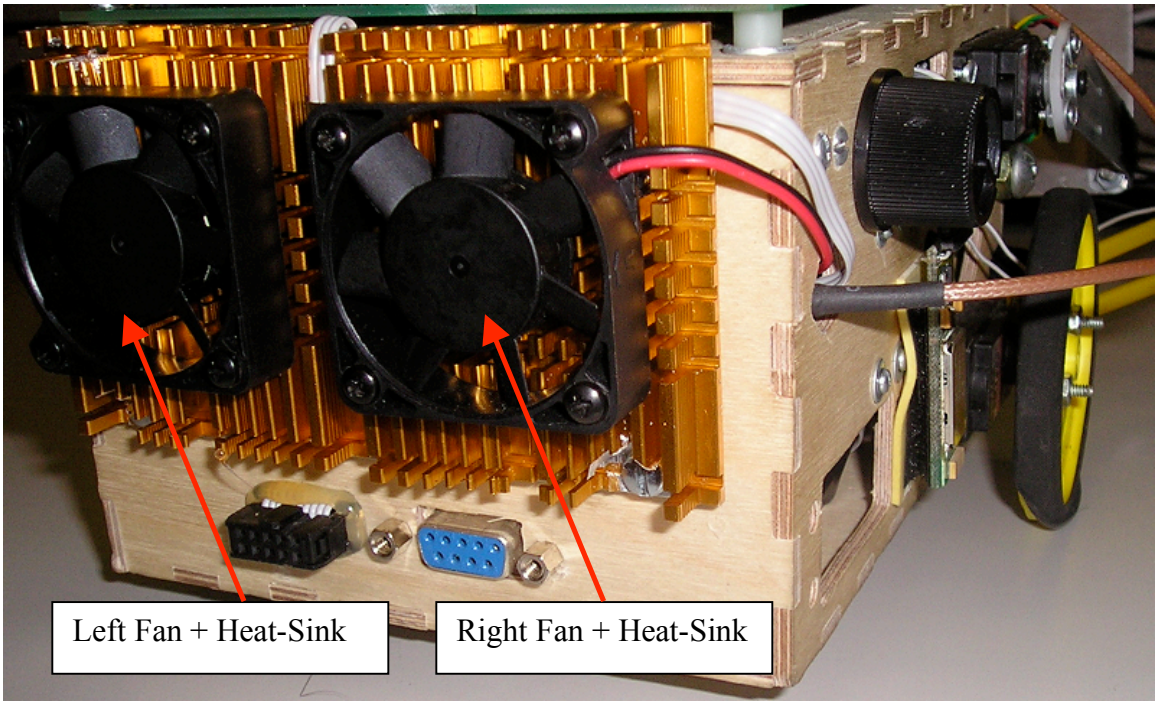
Temperature Sensor Function:

1) int get_temp (); // this is a fast and efficient function at the cost occuracy. The function will return an integer value of the temperature in Celsius.
   Example: temp = get_temp(); // temp has current temperature reading in C



Heat-Sinks and Fans

In addition to the temperature sensor to monitor the amplifier's temperature, two heat-sinks and two fans are located on top of the amplifier to absorb the heat and reduce it to a controlled level.

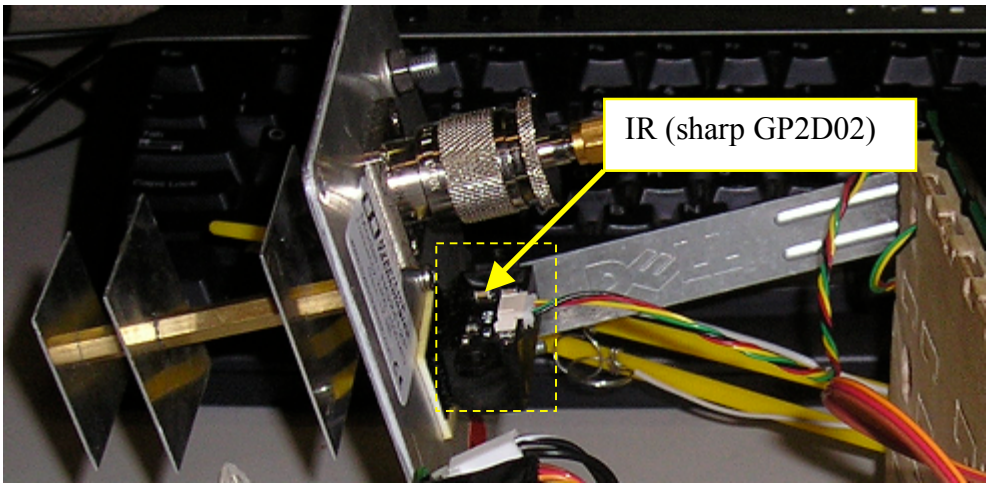


Fan Functions:

1) Void Fan_ON(int fan_num, long time); // turns on specified fan for time (ms)
   Example: Fan_On(left_fan, 4000); // left fan on for 4 second
2) Void Fan_OFF(int fan_num); // turns off specified fan
   Example: Fan_Off(right_fan); // right fan off

Infra-Red (Sharp GP2D02)

The infra red sensor is used for edge of the world detection. It is located on the back of the high power antenna, and would not effect the performance of the antenna due to the grounded plate.

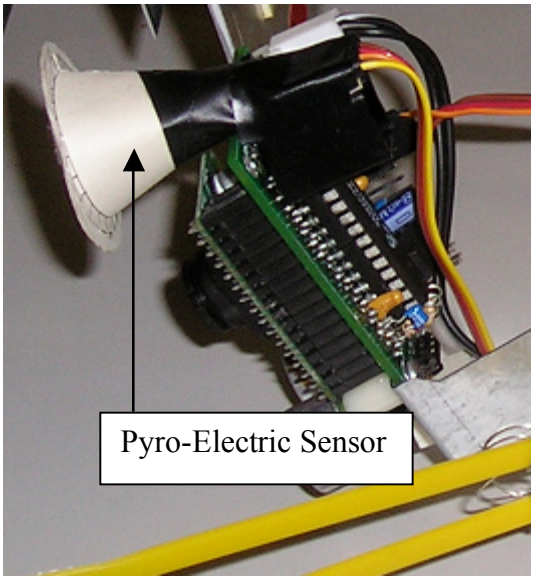


IR Functions:

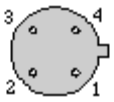
1) int get_ir_reading (); // obtains a vertical distance reading
   Example: height = get_ir_reading();// height has the vertical distance

PyroElectric Sensor

The pyroelectric sensor is used to detect the presence of humans in the path of the radio frequency generated signal. This is a safety issue to ensure a human is not bombarded with high power RF energy.



Bottom View



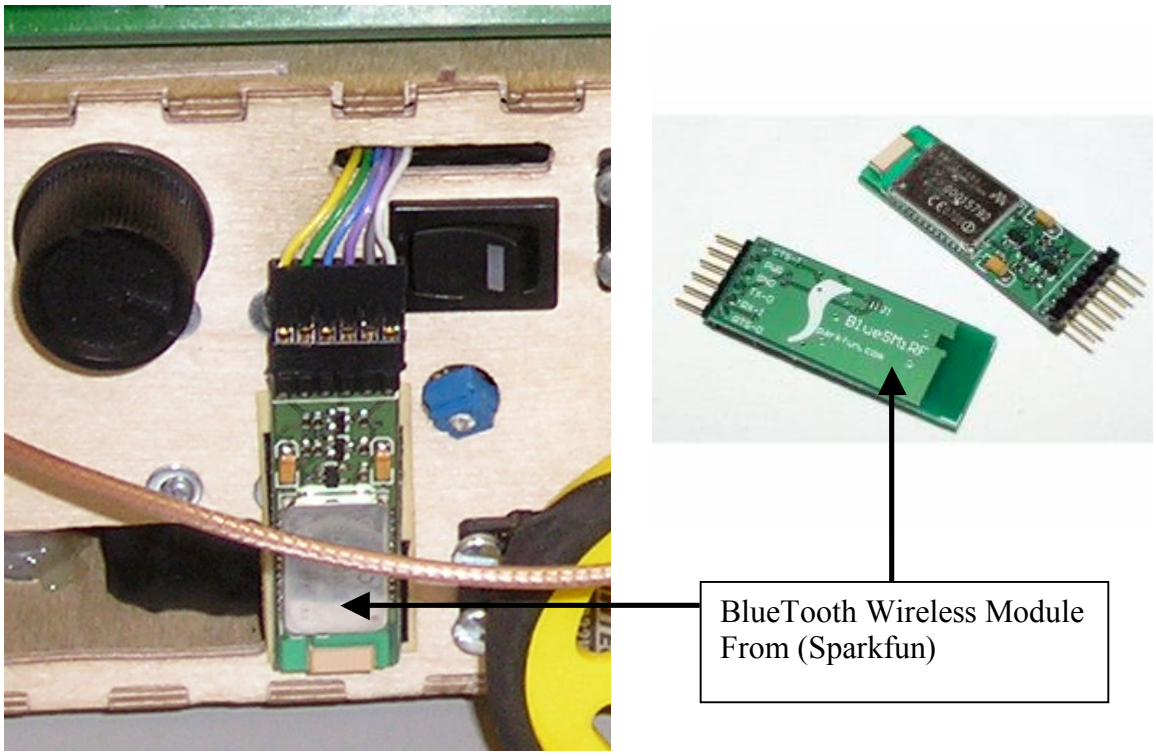
Pin	Description
1	V+
2	Output
3	2.5 V Ref
4	Ground (case)

Pyro-Electric Functions:

1) float get_pyro_voltage();// this will return a floating point voltage reading of the sensor. A 2.5 volts indicates a neutral state (no human movement). A ±0.3 volts change from neutral voltage indicates human movement

BlueTooth Wireless Module

The Bluetooth wireless module is used to communicate with the PC at all times. Sensor information readings are reported back to PC. Also, PC can send back commands to robot. The Bluetooth module has the UART profile loaded into the DSP which allows serial communication with PC at a given agreed baud rate.

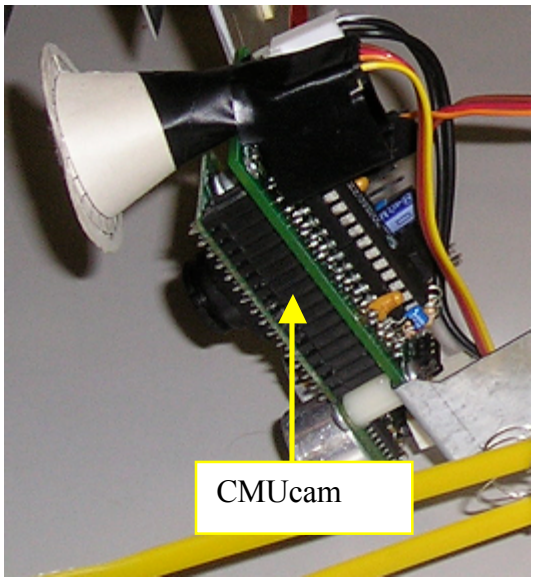


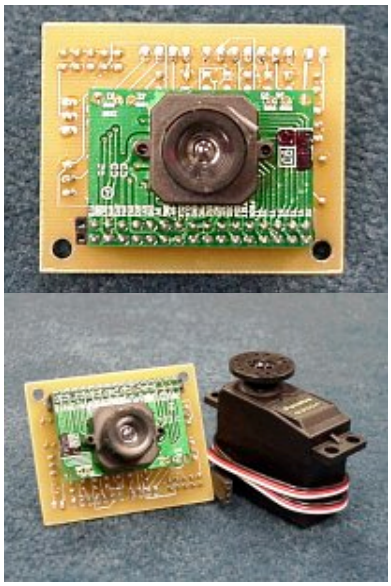
BlueTooth Functions:

1) printf("Hello World, this is BlueTooth Wireless Module"); // all printf statements

CMUCAM Original

The CMUcam is used on Wi-Tesla to detect sensors via colors. When the specified color is captured, Wi-Tesla will follow the color source until it is within 10 cm distance, then it will beam the color source with high radio power beam.





CMUcam Functions:

1) short setup_color (int color);// sets up the cmucam to look for the specified color. Color choices are : Orange, Blue, and Black
   Example: Setup_Color(Orange); // Wi-Tesla would lookout for orange colors

2) short get_color (); //tracks color specified in above function

Conclusion

Wi-Tesla was a great learning experience in the field of sensors and actuators. I've learned a great deal about servos, motors, sensors, and robotic intelligence. Wi-Tesla was a successful project that has met it's objective. Some of the lessons learned in the process of building the robot are summarized below:

1) Wi-Tesla has large power consumption needs.
2) Adding more sensors requires more power
3) The need for more power leads to the addition of more batteries which makes the robot heavier
4) Adding more sensors complicates the design and makes the robot run slower
5) Slower sensor readings leads to missed sensor readings such as collision or edge of the world detection delay
6) Testing and running each sensor alone is the best way to start off
7) Having set functions for each sensor makes the integration process much easier
8) Low battery caused the robot to behave in an unexpected manner

Appendix

Source Code

```
#include <16F777.h>
#device *=16
#device adc=8
#fuses XT,NOWDT,NOBROWNOUT
#use delay(clock=4000000)
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#use rs232(baud=19200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream = cmucam)
#use rs232(baud=9600,parity=N,xmit=PIN_C4,rcv=PIN_C5,bits=8,stream = bluetooth)
#use rs232(baud=19200,parity=N,xmit=PIN_C3,rcv=PIN_C3,bits=8,stream =
servo_control)
//#use rs232(DEBUGGER)
#use fast_io(B)
//#use fast_io(D)
#include <input.c>

//LCD port definition -Done!
```

```
#define gl_dat portb
#define gl_a0  pin_d7 //output pin
#define gl_cs  pin_d6 //output pin
#define gl_wr  pin_d5 //output pin
#define gl_rst pin_d4 //output pin

//Bump switches
#define right_bump  !input(pin_e0) //input bump switch
#define left_bump   !input(pin_e1) //input bump switch
#define RF_ON       output_high(pin_d1)
#define RF_OFF      output_low(pin_d1)
//#define more_bump   !input(pin_e2) //input bump switch

#define pyro_an0 0
#define temp_an1 1

//IR port definitions
#define GP2D02_VIN   pin_d2 //green
#define GP2D02_VOUT  pin_d3 //yellow

//#define gl_rd  pin_d4 // hard wired to VDD=+5V

// servo constatnts
//#define start_byte 0x80
//#define device_id 0x01

//LCD Constants used -Done!
#define SYS_SET         0x40
#define SYS_SLEEP       0x53
#define SYS_CGRAM_ADDR  0x5C
#define SYS_SCROLL      0x44
#define SYS_SCROLL_HDOT 0x5a
#define SYS_SCROLL_RATE 0x5a
#define SYS_CUR_FORM    0x5d
#define SYS_CUR_ADDR    0x46
#define SYS_CUR_READ    0x47
#define SYS_CUR_DIR_RT  0x4c
#define SYS_CUR_DIR_LT  0x4d
#define SYS_CUR_DIR_UP  0x4e
#define SYS_CUR_DIR_DN  0x4f
#define SYS_OVER_LAY    0x5b
#define SYS_MWRITE      0x42
#define SYS_MREAD       0x43

#define LCD_DISP_OFF    0x58
#define LCD_DISP_ON     0x59
```

```
#define LCD_INVERSE     0xff
#define LCD_NORMAL      0x00
#define LCD_CR          0x20    //32 Chars/bytes per line
#define LCD_W           0x100   //256 pixels wide
#define LCD_H           0x80    //128 pixels high
#define LCD_Lh          0x08    //Height of line (8x8 characters)
#define LCD_L1          0x00    //Line 1
#define LCD_L2          0x20    //Line 2
#define LCD_L3          0x40    //Line 3
#define LCD_L4          0x60    //Line 4
#define LCD_L5          0x80    //Line 5
#define LCD_L6          0xA0    //Line 6
#define LCD_L7          0xC0    //Line 7
#define LCD_L8          0xE0    //Line 8
#define LCD_L9          0x100   //Line 9
#define LCD_L10         0x120           //Line 10
#define LCD_L11         0x140           //Line 11
#define LCD_L12         0x160           //Line 12
#define LCD_L13         0x180           //Line 13
#define LCD_L14         0x1A0           //Line 14
#define LCD_L15         0x1C0           //Line 15
#define LCD_L16         0x1E0           //Line 16
#define LCD_GRH         0x1000  //Graphic home position

//#define button_down   !input(pin_c4)
//#define eeprom_size   256
#define echo            pin_c1
#define pulse_trigger   pin_c0
#define trigger_delay   10      // 10ms ultrasonic reset
#define out_of_range    0xffff


// servo control constants
#define servo_reset  pin_c2
#define right_servo  0
#define left_servo   1
#define right_arm    2
#define left_arm     3
#define neutral_speed 75

#define full_speed   25
#define speed5       25
#define speed4       20
#define speed3       15
#define speed2       10
#define speed1       5
```

```
#define speed0       0
#define stop_speed   0

#define turn_delay   500
//servo constants:
#define start_byte      0x80
#define device_id       0x01
#define go_forward      0
#define go_backward     1
#define repeat_command  3        //servo repeat command
#define turn_time       5       // turns 5 *(100ms) = 0.5sec


//CMUcam constants:
#define orange 0
#define black  1

//#define average_color   4
//servo global variables
int servo_command;
int servo_num;
int data_1 ;
int data_2 ;

int counter0_overflow;
int counter1_overflow;
int counter2_overflow;

short timeout_error;

//long cmu_param[8];
int mx,my,x1,y1,x2,y2,pixels,confidence;

// graphic LCD
char line_string [33];
void gl_init();
void gl_grfclr();
void gl_grfhome();
void gl_strobe();
void gl_txtclr();
void gl_setaddr   (int gl_cur, long gl_addr);
void send_data    (int gl_byte);
void send_command (int gl_cmd);
#separate
void g_printf     (char string[],long Line_num);
void g_clear_line (long Line_num);
```

```
//void move_forward    (int servo_num, int speed); //replaced with move
//void move_backward   (int servo_num, int speed); //replaced with move

void left_turn       (int speed);
void right_turn      (int speed);

//void right_arm_turn  (int angle); //replaced with turn_arm
//void left_arm_turn   (int angle); //replaced with turn_arm

void turn_off        (int servo_num); //optimized
void turn_on         (int servo_num); //optimized

void forward         (int speed,int time); //avoid using this
void backward        (int speed,int time); // avoid using this

void move            (int direction, int servo_num, int speed);
void send_servo_data1(void);
void send_servo_data2(void);
void turn_arm        (int arm, int angle);

void check_bumps     (void);

float get_distance   ();
float get_average_distance ( int num_average);

//char  timed_getc     ();
//short rs232_wait     (char wait_char,int time_out);
#separate
short setup_color    (int color);
#separate
short get_color      ();
#separate
int   get_ir_reading ();
int   get_temp       ();

float get_pyro_voltage();

int get_temp()
{
  set_adc_channel(temp_an1);
  delay_ms(1);
  return (read_adc()<<1);
}

float get_pyro_voltage()
```

```
{
  //int pyro_adc;

  set_adc_channel(pyro_an0);
  delay_ms(1);
  //pyro_adc = read_adc();                          //get pyro reading
  //pyro_volts = pyro_adc * (5.00/256);
  return (read_adc() * 0.02);

}

void turn_arm(int arm,int angle)
{
  int angle_rotation;
  servo_command = 3;
  servo_num = arm;


  if (angle > 210)
  {
    if (arm==left_arm)
    {
      angle_rotation = 0;     //left arm
    }
    else angle_rotation = 255; //right arm
  }
  else angle_rotation = angle + (angle>>2);      // angle_rotation(max)=255 <==>
angle(max)=210 deg .. approx 1+(1/4)

  if (arm==left_arm) angle_rotation = 255-angle_rotation;

  data_1 = bit_test (angle_rotation,7);
  bit_clear(angle_rotation,7);
  data_2 = angle_rotation;

send_servo_data2();
}

void send_servo_data1()
{
int i;
     for (i=0;i<repeat_command;i++)
     {
     fprintf(servo_control,
"%c%c%c%c%c",start_byte,device_id,servo_command,servo_num,data_1); //send data
to servo
```

```
    }
}
void send_servo_data2()
{
int i;
    for (i=0;i<repeat_command;i++)
    {
    fprintf(servo_control,
"%c%c%c%c%c%c",start_byte,device_id,servo_command,servo_num,data_1,data_2);
//send data to servo
    }
}

void move(int direction,int servo_number, int speed)
{

  servo_command = 2;
  servo_num = servo_number;

                if(direction==go_forward)
                {
                        if (servo_num==right_servo)
                        {
                                data_1 = neutral_speed + speed;                 //right
servo forward
                        }
                        else data_1 = neutral_speed - speed;  //left servo forward
                }
                else // direction is backward
                {
                        if (servo_num==right_servo)
                        {
                                data_1 = neutral_speed - speed;                 //right
servo backward
                        }
                        else data_1 = neutral_speed + speed; //left servo backward
                }
    send_servo_data1();
}
#separate
int get_ir_reading() {
        int counter=9;
        int reading=0;

        output_low(GP2D02_VIN);   // start cycle with Vin low
        delay_ms(1);              // give the sensor a little time before we bug it
```

```
        while (!input(GP2D02_VOUT));       //wait for Vout to go high

        do {
                delay_cycles(4);         // minimum wait is 2us, max is 170us, 4 worked
    output_low(GP2D02_VIN);  // tell the sensor that we want a bit
                delay_cycles(2);         // might be as low as 1 (maybe), 2 worked
                reading=reading<<1;  //left shift the reading value
                reading=reading|(input(GP2D02_VOUT)); // put the newest reading into
the
                                                // LSB of reading
                output_high(GP2D02_VIN); // we read the bit, and get ready for the next
one
                counter--;
        } while (counter>0);

        // We leave the Vin pin high after finishing the reading
        // This resets the sensor in order for a new reading next
        // time

        // An 8 bit number indicating the distance should now be
        // sitting in the variable 'reading'

        return reading ;
}
#include <graphical_LCD.h>

//#include <timer_interrupts.h>

#include <check_bumps().h>

#include <void_move_forward(int_servo_num_int_speed).h>
#include <void_move_backward(int_servo_num_int_speed).h>

//#include <void_forward(int_speed_int_time).h>
//#include <void_backward(int_speed_int_time).h>

#include <void_left_Turn(int_speed_int_time).h>
#include <void_right_Turn(int_speed_int_time).h>

//#include <void_right_arm_turn(int_angle).h>
//#include <void_left_arm_turn(int_angle).h>

#include <void_turn_off(int_servo_num).h>
#include <void_turn_on(int_servo_num).h>

#include <commented_float_get_distance().h>
```

```
#include <float_get_distance().h>
#include <float_get_average_distance(int_num_average).h>


//#include <char_timed_getc().h>
//#include <short_rs232_wait(int_wait_char_int_time_out).h>

#include <short_setup_color().h>
#include <short_get_color().h>

////////////////////////////MAIN()/////////////////////////////////////////////
void main() {

//LCD Global Variables used -Done!
int  gl_cmd;
int  gl_cur;
int  gl_byte;
int  gl_i;
int  gl_j;
int  gl_k;
long gl_x;
long gl_y;
long gl_addr;
long gl_w;
int  gl_read;
int  gl_old;
long pix_x;
long pix_y;
long gl_addrlo;
long gl_addrhi;
int  gl_nib;

// servo control variables
int servo_num;
int right_arm_angle;
int left_arm_angle;

//general variables
int i=0,j=0,k=0;

//variables used for UltraSound
long returned_distance;
float distance;

//variables used for ADC
int pyro_adc;      //16-bit integers
```

```
float pyro_volts;
int temp_cel; //should be long, but doubt temperature would be that high
float temp_volts;

// CMUcam variables
char cmu_char[10];
long cmu_param[8];

int ir_reading;
int speed;
char bt_rx[33];
char rx_data[5];
char gl_rx;
int junk;
  //some stuff
  setup_spi(FALSE);
  //setup_counters(RTCC_INTERNAL,RTCC_DIV_2);
  //setup_timer_0(RTCC_INTERNAL| RTCC_DIV_8);
  //setup_timer_0(RTCC_INTERNAL| RTCC_DIV_64);   //timer_0 will overflow every
16.32 msec
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_128);   //timer_0 will overflow every
32.7 msec, resolution = 128us (increments every 128us)
  //setup_timer_1(T1_DISABLED);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);      //overflows every 65.5ms
  setup_timer_2(T2_DIV_BY_16,255,16);          //overflows every 65.5ms
  //setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);

  set_timer0(0);                    //reset timer
  set_timer1(0);                    //reset timer
  set_timer2(0);                    //reset timer

  //counter0_overflow = 0;
  //counter1_overflow = 0;
  //counter2_overflow = 0;

  set_tris_b(0x00);
  //set_tris_d(0x00);

  //enable_interrupts(INT_TIMER0);          //activate timer
  //enable_interrupts(INT_TIMER1);          //activate timer
  //enable_interrupts(INT_TIMER2);          //activate timer

  //disable_interrupts(INT_TIMER0);         //dectivate timer
  //disable_interrupts(INT_TIMER1);         //dectivate timer
```

```
  //disable_interrupts(INT_TIMER2);          //dectivate timer

  //enable_interrupts(GLOBAL);               //activate global interrupt flag

  RF_OFF; //turn off RF

  //LCD
  fprintf(bluetooth,"\r\nInitializing Graphical LCD...");
  output_low(gl_rst);
  delay_ms(100);
  output_high(gl_rst); //Make sure reset is high
  delay_ms(100);
  output_low(gl_cs); //Chip select active
  delay_ms(10);
  gl_init();                    //Initialize display
  fprintf(bluetooth,"Done!\r\n");

  strcpy(line_string, "Welcome to IMDL2006 & Wi-Tesla !");
  //g_printf(line_string,LCD_L1);

  delay_ms(10);

  //setup ADC channels and clk
  strcpy(line_string, "Initializing ADC channels...");
  //g_printf(line_string,LCD_L2);
  //fprintf(bluetooth,"Initializing ADC channels...");
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(AN0_TO_AN1);
  //fprintf(bluetooth,"Done!\r\n");


  strcpy(line_string, "communicating with CMUcam...");
  //g_printf(line_string,LCD_L3);
  fprintf(bluetooth,"%s",line_string);
  setup_color(black); //need to setup color everytime you change color parameter
  get_color();
  fprintf(bluetooth,"Done!\n\r");

  sprintf(line_string, "CMUcam color captured !");
  //g_printf(line_string,LCD_L4);
  fprintf(bluetooth,"%s\n\r",line_string);

  sprintf(line_string, "%u %u %u %u %u %u %u
%u",mx,my,x1,y1,x2,y2,pixels,confidence);
  //g_printf(line_string,LCD_L5);
  fprintf(bluetooth,"%s\n\r",line_string);
```

```
  // ultrasonic initialize
  //fprintf(bluetooth,"\n\rinitializing ultrasonic...");
  sprintf(line_string, "initializing ultrasonic...");
  //g_printf(line_string,LCD_L6);
  fprintf(bluetooth,"%s",line_string);
  output_low(pulse_trigger);
  fprintf(bluetooth,"Done!\n\r");

  //fprintf(bluetooth,"\n\rresetting servo controller...");
  sprintf(line_string, "resetting servo controller...");
  //g_printf(line_string,LCD_L7);
  fprintf(bluetooth,"%s",line_string);
  output_low(servo_reset);
  delay_ms(500);
  output_float(servo_reset);
  delay_ms(500);
  fprintf(bluetooth,"Done!\n\r");

  //turn_on(right_servo);
  //turn_on(left_servo);

  //turn_on(right_arm);
  //turn_on(left_arm);

  //fprintf(bluetooth,"\n\rright arm turnning");
  //sprintf(line_string, "right arm turnning");
  //g_printf(line_string,LCD_L8);

  //right_arm_turn(180);
  //delay_ms(500);
  //turn_off(right_arm);
/*

  for ( k=220;k>120;k--)     //lift up right arm then put it back down
    {
      //right_arm_turn(k);
      turn_arm(right_arm,k);
      delay_ms(10);
    }
  for ( k=120;k<220;k++)
    {
      //right_arm_turn(k);
      turn_arm(right_arm,k);
      delay_ms(10);
    }
```

```
  turn_off(right_arm);
  delay_ms(10);

    k=90;
    for ( k=90;k<180;k++)
    {
      //left_arm_turn(k);
      turn_arm(left_arm,k);
      delay_ms(10);
    }
*/
//left_arm_turn(210);
//delay_ms(1000);
//turn_off(left_arm);
  //delay_ms(5000);
  //left_arm_turn(180);


  //sprintf(line_string, "setting up CMUcam ...");
  //g_printf(line_string,LCD_L8);
  //setup_color();
  //servo_command = 1;
  //servo_num = left_arm;
  //data_1 = 32;

  fprintf(servo_control, "%c%c%c%c%c",start_byte,device_id,1,left_arm,32); //send data
to servo
  delay_ms(10);
  fprintf(servo_control, "%c%c%c%c%c",start_byte,device_id,1,right_arm,32); //send
data to servo
  delay_ms(10);
  fprintf(servo_control, "%c%c%c%c%c",start_byte,device_id,1,left_servo,64); //send
data to servo
  delay_ms(10);
  fprintf(servo_control, "%c%c%c%c%c",start_byte,device_id,1,right_servo,64); //send
data to servo
  delay_ms(10);

  //turn_arm(left_arm,180);

  setup_color(orange);
  strcpy(rx_data,"exit");
  //gl_byte=0;
  k=0;
  while(true)
{
```

```
loop1:

  send_command(SYS_CUR_ADDR);         //CSRW command
        send_data(0x00);
        send_data(0x10);
        send_command(SYS_CUR_DIR_RT);         //Cur movement right
        send_command(SYS_MWRITE);

  while(fgetc(bluetooth)!=']')
  // after the ']' microcontroller will lose 31 characters (because of software rs232) to
initiate i and j
  for(i=0;i<128;i++)   //128/8=16
    {
      for(j=0;j<32;j++) //256/8=32
      {
        send_data(fgetc(bluetooth));
      }
    }


  goto loop1;
  //while(fgetc(bluetooth)!=']');
  //fgetc



  left_arm_angle = 180;
  turn_arm(left_arm,left_arm_angle);

  check_bumps(); //checks right and then left bump, and turns 0.5 sec opposite to bump

  //move_forward(right_servo, speed3);
  //move_forward(left_servo, speed3);

  move(go_forward,right_servo,speed5);
  move(go_forward,left_servo,speed5);

  ir_reading= get_ir_reading();
  sprintf(line_string, "IR Reading=%u",ir_reading);
  g_printf(line_string,LCD_L8);
  fprintf(bluetooth,"%s\n\r",line_string);

  if(ir_reading<180)
  {
  //backward(speed5,5);
  move(go_backward,right_servo,speed3);
```

```
move(go_backward,left_servo,speed3);
delay_ms(500);

//left turn
left_turn(speed5);
delay_ms(500);
move(go_forward,right_servo,speed5);
move(go_forward,left_servo,speed5);

//left_turn   (speed5,5);
}

check_bumps();

pyro_volts = get_pyro_voltage();
sprintf(line_string, "Pyro Reading=%1.2f volts",pyro_volts);
g_printf(line_string,LCD_L9);                    //display voltage
fprintf(bluetooth,"%s\n\r",line_string);

check_bumps();

temp_cel = get_temp();
//set_adc_channel(temp_an1);
//delay_ms(1);
//temp_cel = read_adc()<<1;
sprintf(line_string, "Temperature= %u%cC",temp_cel,167);
g_printf(line_string,LCD_L10);
fprintf(bluetooth,"%s\n\r",line_string);


check_bumps(); //checks right and then left bump, and turns 0.5 sec opposite to bump

//distance = get_average_distance(10);          //get 10 averages

distance = get_distance();          //get 10 averages
sprintf(line_string, "Distance=%3.2f cm",distance);   //display average distance
g_printf(line_string,LCD_L11);
fprintf(bluetooth,"%s\n\r",line_string);

if (distance < 30 )  //if less than 30cm then turn
{
  right_turn(speed5); //right turn
  delay_ms(turn_delay); //for half second
  move(go_forward,right_servo,speed3);
  move(go_forward,left_servo,speed3);
}
```

```
//setup_color(orange);

getcolor:

  get_color();

  //sprintf(line_string, "%u %u %u %u %u %u %u
%u",mx,my,x1,y1,x2,y2,pixels,confidence);
  //g_printf(line_string,LCD_L5);

  //if ceter is off to the left, then turn left
  if ( mx!=0 && my!=0)
  {
  /*
    distance = get_distance();                //get 10 averages
    //sprintf(line_string, "Distance=%3.2f cm",distance);   //display average distance
    //g_printf(line_string,LCD_L11);                        //

    if (distance < 15 && confidence>40)  //if less than 5cm then backoff
    {


      left_turn   (speed2);
      delay_ms(turn_delay);

      turn_off(right_servo);
      turn_off(left_servo);

      RF_ON;
      delay_ms(8000);
      RF_OFF;

      move(go_backward,right_servo,speed3);
      move(go_backward,left_servo,speed3);
      delay_ms(500);

      //left turn
      left_turn(speed5);
      delay_ms(500);
      move(go_forward,right_servo,speed3);
      move(go_forward,left_servo,speed3);

    }
```

```
    else if (distance < 50 )
    {
      speed = speed1;
    }
    else if (distance < 100 )
    {
      speed = speed2;
    }
    else
    {
      speed = speed3;
    }
*/
     //sprintf(line_string, "color detected, mx=%u, my=%u",mx,my);
     //g_printf(line_string,LCD_L12);

      if (mx>=30 && mx <=50 && my>=40 && my<=100)
      {
        // move forward (center found)
        move(go_forward,right_servo,speed2);
        move(go_forward,left_servo,speed2);

      }
      else
      {
        if ( mx<40 && mx>0)
        {
          //tilt down
          left_arm_angle =left_arm_angle+4;
          turn_arm(left_arm,left_arm_angle);
        }
        else
        {
          //tilt up
          left_arm_angle =left_arm_angle-4;
          turn_arm(left_arm,left_arm_angle);
        }
        if ( my<70 && my>0)
        {
          //right turn
          move(go_backward,right_servo,speed1);
          move(go_forward,left_servo,speed1);

        }
        else
        {
```

```
            //left turn (robot has hard time doing this)
            move(go_forward,right_servo,speed2);
            move(go_backward,left_servo,speed2);

          }
        }

    goto getcolor;
  }

}
}

////////////////////////////g_printf(char line_in)//////////////////////////////////
#separate
void g_printf(char string[],long Line_num)
{
    int i;

    gl_setaddr(SYS_CUR_DIR_RT,line_num);       // (curson direction, address)
    send_command(SYS_MWRITE);
    for (i=0;i<strlen(string);i++)
    {
     send_data(string[i]); // i don't have to cast a character (int)string to send to the port
    }
}
////////////////////////////g_clear_line(char line_in)//////////////////////////////////
void g_clear_line(long Line_num)
{
    int i;

    gl_setaddr(SYS_CUR_DIR_RT,line_num);       // (curson direction, address)
    send_command(SYS_MWRITE);
    for (i=0;i<32;i++)
    {
     send_data(0x20);//ASCII Space
    }
}
////////////////////////////gl_init()//////////////////////////////////
void gl_init() //initialize graphical LCD
  {
  int i;            //  P1   P2   P3   P4   P5   P6   P7   P8
  int sys_set_array[8]={0x30,0x87,0x07,0x1f,0x52,0x7f,0x20,0x00};
  int sys_scroll_array[6]={0x00,0x00,0x7f,0x00,0x10,0x7f};

  //system set
```

```
  send_command(SYS_SET);
  //fprintf(rs232,"initializing lcd ..\n\r");
        for (i=0;i<8;i++)
                {
    send_data(sys_set_array[i]);
    }
        //scroll
        send_command(SYS_SCROLL);
  for (i=0;i<6;i++)
  {
  send_data(sys_scroll_Array[i]);
  }
        //Hdot
        send_command(SYS_SCROLL_RATE);
  send_data(0x00);
        //Overlay
        send_command(SYS_OVER_LAY);
        send_data(0x01);
        //DISP ON
        //send_command(LCD_DISP_OFF);
        send_command(LCD_DISP_ON);
  send_data(0b00010111);
        //Clear Graphic screen
        gl_grfclr();
        //Clear Text screen
        gl_txtclr();
        //CSRW set cursor home
        send_command(SYS_CUR_ADDR);
  send_data(0x00);
  send_data(0x00);
        send_command(SYS_CUR_DIR_RT);
  send_command(SYS_MWRITE);
  send_data(0x00);              //start with offset
        //CSR FORM
        send_command(SYS_CUR_FORM);
        send_data(0x04);
        send_data(0x86);
        //DISP ON
        send_command(LCD_DISP_ON);
        send_data(0b00010100);        //cursor off
        //CSR DIR
        send_command(SYS_CUR_DIR_RT);
  }

//////////////////////////gl_grfclr()///////////////////////////////////
 void gl_grfclr() //Clear Graphic Screen
```

```
  {
  int i,j;
  send_command(SYS_CUR_ADDR);        //CSRW command
        send_data(0x00);
        send_data(0x10);
        //send_data(0x00);
        send_command(SYS_CUR_DIR_RT);        //Cur movement right
        send_command(SYS_MWRITE);
        //not sure id 128 is required
  for(i=0;i<128;i++)   //128/8=16
  {
    for(j=0;j<32;j++) //256/8=32
    {
      send_data(LCD_NORMAL);
    }
  }
        send_command(SYS_CUR_ADDR);//CSRW command
        send_data(0x00);
  send_data(0x00);
  //send_data(0x10)
        send_command(SYS_CUR_DIR_RT);        //Cur movement right

  }
//////////////////////////gl_grfhome()////////////////////////////////
void gl_grfhome()
  {
        send_command(SYS_CUR_ADDR);//CSRW command
        send_data(0x00);
  send_data(0x10);
        send_command(SYS_CUR_DIR_RT);        //Cur movement right
  }
//////////////////////////gl_strobe()////////////////////////////////
void gl_strobe()
  {
        output_low(gl_cs);              //Chip select active
  output_low(gl_wr);
  delay_cycles(1);       //same as nop
  output_high(gl_wr);
  output_high(gl_cs);    //Chip select de-active
  }
//////////////////////////gl_txtclr()////////////////////////////////
void gl_txtclr()
  {
  int i,j;
        send_command(SYS_CUR_ADDR);//CSRW command
        send_data(0x00);
```

```
  send_data(0x00);
        send_command(SYS_CUR_DIR_RT);         //Cur movement right
        send_command(SYS_MWRITE);
        for (i = 0;i<16;i++)
          {
     for (j=0;j<32;j++)
     {
                  send_data(0x20);    //ASCII Space
     }
     }
        send_command(SYS_CUR_ADDR);//CSRW command
        send_data(0x00);
  send_data(0x00);
        send_command(SYS_CUR_DIR_RT);         //Cur movement right
  }
////////////////////////////gl_setaddr()/////////////////////////////////////
//Routine to set the address
//assumes address is in gl_addr and cursor movment in gl_cur
void gl_setaddr(unsigned gl_cur, long gl_addr)
  {
        //CSRW set cursor home
        send_command(SYS_CUR_ADDR);      //Address command
                        //Vram text memory starts at $0000 and graphics at $1000
        send_data(make8(gl_addr,0));     //lower byte
        send_data(make8(gl_addr,1));     //upper byte
        send_command(gl_cur);           //Set cursor movement to left,right,up or down

  }
////////////////////////////send_data()/////////////////////////////////////
void send_data(unsigned gl_byte) //done!
  {
  output_low(gl_a0);
        output_b(gl_byte);
        gl_strobe();
  }
////////////////////////////send_command()/////////////////////////////////////
void send_command(unsigned gl_cmd) //done!
  {
  output_high(gl_a0);
        output_b(gl_cmd);
  gl_strobe();
  }
//////////////////////////// check bumps and act/////////////////////////////////
void check_bumps()
{
  if (right_bump)
```

```
  {
  left_turn(speed5); //left turn half second
  delay_ms(turn_delay);
  move(go_forward,right_servo,speed3);
  move(go_forward,left_servo,speed3);
  }

  if (left_bump)
  {
  right_turn(speed5); //right turn half second
  delay_ms(turn_delay);
  move(go_forward,right_servo,speed3);
  move(go_forward,left_servo,speed3);
  }
}
void move_forward(int servo_num, int speed)
  {

  int servo_command = 2,i;


    if      (servo_num == right_servo) data_1 = neutral_speed + speed;
    else if  (servo_num == left_servo)  data_1 = neutral_speed - speed;
    else     data_1 = neutral_speed;

    for (i=0;i<repeat_command;i++)
    {
    fprintf(servo_control,
"%c%c%c%c%c",start_byte,device_id,servo_command,servo_num,data_1); //send data
to servo
    }
    //fprintf(rs232, "\n\rmoving forward: cmd=%d, d1=%d, servo=%d,
speed=%d",servo_command,data_1,servo_num,speed);

  }
void move_backward(int servo_num, int speed)
  {

  int servo_command = 2,i;


    if      (servo_num == right_servo) data_1 = neutral_speed - speed;
    else if  (servo_num == left_servo)  data_1 = neutral_speed + speed;
    else     data_1 = neutral_speed;

    for (i=0;i<repeat_command;i++)
```

```
      {
      fprintf(servo_control,
"%c%c%c%c%c",start_byte,device_id,servo_command,servo_num,data_1); //send data
to servo
      }
      //fprintf(rs232, "\n\rmoving forward: cmd=%d, d1=%d, servo=%d,
speed=%d",servo_command,data_1,servo_num,speed);

   }
void left_turn(int speed)
{
 move(go_forward, right_servo, speed);
 move(go_backward, left_servo, speed);

/*
   int i;

   turn_on(right_servo);
   turn_on(left_servo);

   move_forward(right_servo, speed);
   move_backward(left_servo,speed);

      for (i=0;i<time;i++)
      {
        delay_ms(100);
      }
*/

/*
   move_forward(right_servo, speed0); //neutralize servos
   move_forward(left_servo, speed0);  //neutralize servos

   turn_off(right_servo);
   turn_off(left_servo);
*/

}
void right_turn(int speed)
{
 move(go_backward, right_servo, speed);
 move(go_forward, left_servo, speed);
   //int i;

   //turn_on(right_servo);
   //turn_on(left_servo);
```

```
  //move_backward(right_servo, speed);
  //move_forward(left_servo,speed);


    //for (i=0;i<time;i++)
    //{
    //   delay_ms(100);
    //}
/*
  move_forward(right_servo, speed0); //neutralize servos
  move_forward(left_servo, speed0);  //neutralize servos


  turn_off(right_servo);
  turn_off(left_servo);
*/
}
void right_turn(int speed)
{
 move(go_backward, right_servo, speed);
 move(go_forward, left_servo, speed);
  //int i;

  //turn_on(right_servo);
  //turn_on(left_servo);

  //move_backward(right_servo, speed);
  //move_forward(left_servo,speed);


    //for (i=0;i<time;i++)
    //{
    //   delay_ms(100);
    //}
/*
  move_forward(right_servo, speed0); //neutralize servos
  move_forward(left_servo, speed0);  //neutralize servos


  turn_off(right_servo);
  turn_off(left_servo);
*/
}
void turn_on(int servo_num)
{
```

```
    //int servo_command = 0,i;
    servo_command = 0;
    servo_num=servo_num;
    data_1 = 0x4f; //servo on

    send_servo_data1();

    //for (i=0;i<repeat_command;i++)
    //{
    //fprintf(servo_control,
"%c%c%c%c%c",start_byte,device_id,servo_command,servo_num,data_1); //send data
to servo
    //}

}
void turn_off(int servo_num)
{
    //int servo_command = 0,i;
    servo_command = 0;
    servo_num=servo_num;
    data_1 = 0x0f; //servo off

    send_servo_data1();

    //for (i=0;i<repeat_command;i++)
    //{
    //fprintf(servo_control,
"%c%c%c%c%c",start_byte,device_id,servo_command,servo_num,data_1); //send data
to servo
    //}
}
float get_distance()
  {
  //float echo_delay;
  float distance;

  output_high(pulse_trigger);
  delay_us(trigger_delay);
  output_low(pulse_trigger);

  while (!input(echo)); //do nothing
  // low to high detected, reset timer
  set_timer1(0);
  while (input(echo)); //wait for High to Low
  //echo_delay = get_timer1();
  distance = get_timer1()/58;
```

```
  //if (echo_delay>40000)
  //{
  //distance = out_of_range;
  //distance = 0;
  //}
  //else
  //{
  // distance = echo_delay/58; // div by 29 for round distance in cm, 58 is one way
  //}
  return distance;

  }
float get_average_distance ( int num_average)
{
 float distance =0 ;
 long returned_distance = 0;
 int average_scale = 0, i;


       for (i=0;i<num_average;i++) //get 10 average distances
       {
       returned_distance = get_distance();

         if (returned_distance > 0)            //object within range
         {
         average_scale = average_scale +1;
         distance = distance + returned_distance;
         }

       delay_ms(20);
       }
       distance = distance /average_Scale;

       return distance;

       //sprintf(line_string, "Distance=%3.2f cm, counter= %u",distance,k);
       //fprintf(rs232,"\n\rDistance=%3.2f cm",distance);
       //g_printf(line_string,LCD_L2);
}
////////////////////////////setup_color( )////////////////////////////////////
#separate
short setup_color( int color)
{

  fprintf(cmucam,"PM 1\r");     //poll mode active
  while(getc()!=':');
```

```
  fprintf(cmucam,"MM 1\r");     //middle mass mode
  while(getc()!=':');

  switch (color) {
   case 0: fprintf(cmucam,"TC 150 240 35 85 16 20\r");  //Orange
          while(getc()!=':');
          break;

   case 1: fprintf(cmucam,"TC 30 45 30 40 25 35\r");  //Black
          while(getc()!=':');
          break;
   default: return false;
          break; }

  fprintf(cmucam,"RM 3\r");     //raw mode active
  while(getc()!=':');
  return true;
}
////////////////////////////color_found( )///////////////////////////////////
#separate
short get_color( )
{

  // int i,j;
   //char cmu_char[8];
/*
   fprintf(rs232,"PM 1\r");     //poll mode active
   while(getc()!=':');
   fprintf(rs232,"MM 1\r");     //middle mass mode
   while(getc()!=':');
   fprintf(rs232,"TC 200 240 50 70 15 20\r");
   while(getc()!=':');
   fprintf(rs232,"RM 3\r");     //raw mode active
   while(getc()!=':');
*/
//mx=0,my=0,x1=0,y1=0,x2=0,y2=0,pixels=0,confidence=0;
//for (j=0;j<4;j++)
//   {
//      delay_ms(1);              //somehow this is a needed delay ..

      fprintf(cmucam,"TC\r");     //track color
      while(getc()!='M');

      mx          =getc();
      my          =getc();
      x1          =getc();
```

```
    y1          =getc();
    x2          =getc();
    y2          =getc();
    pixels      =getc();
    confidence  =getc();

    return true;
    //if ( mx && my && x1 && y1 && x2 && y2 && pixels && confidence ) return
false;
    //else return true;
/*
    for (i=0;i<8;i++)
    {
      cmu_char[i]=getc();
    }

    mx          =mx + cmu_char[0];
    my          =my + cmu_char[1];
    x1          =x1 + cmu_char[2];
    y1          =y1 + cmu_char[3];
    x2          =x2 + cmu_char[4];
    y2          =y2 + cmu_char[5];
    pixels      =pixels     + cmu_char[6];
    confidence  =confidence + cmu_char[7];
  }
    mx          = mx>>2;
    my          = my>>2;
    x1          = x1>>2;
    y1          = y1>>2;
    x2          = x2>>2;
    y2          = y2>>2;
    pixels      = pixels>>2;
    confidence  = confidence>>2;
*/

    //return true;
}
```