University of Florida
Department of Electrical and Computer Engineering
EEL5666
Intelligent Machines Design Laboratory

BeggarBot
Final Report

Date:	8/8/02
Student Name:	David Choy
TA:	Tae Choi
	Uriel Rodriguez
Instructor:	Arroyo/Schwartz

Table of Contents

Abstract

BeggarBot is an autonomous robot that searches for humans in order to ask them for money. If a donation is made, BeggarBot is happy and goes along its way looking for more humans. If a donation is not made, BeggarBot gets angry and will shoot the stingy person. As a result of being lazy, BeggarBot will stop all actions once it has collected enough money for the day.

Executive Summary

BeggarBot performs all behaviors as intended in the initial design proposal. I completed the robot about one week in advance, which gave me time to add the ability to talk. Although this was successful, I did not have a housing for the speech board and speaker on BeggarBot. This explains why the speech board looks like it was thrown on the side of the robot - because it really was. One other improvement I could have made to the robot's speech is to amplify the output to the speakers. However, I did not have enough free time to devote to this. Another issue that can be improved upon is the selection of the pyroelectric sensor. Although the sensor I selected was one of the more expensive ones, it provided inconsistent data. If this robot were to be used for a more serious application, a more accurate pyroelectric sensor is required. Overall I am satisfied with the final product I have achieved and believe my robot is a success.

Introduction

Everyone has been approached at least once in their lives by people asking for money, whether it be a family member, friend, or even a stranger. I will refer to people who ask for money as beggars. In many instances, the beggar provides an honest reason for needing the money - usually the case for family and friends. Strangers, on the other hand, may or may not be telling the truth. I have noticed in my experience that some dishonest people actually make a living asking for money based on a few colorful lies they circulate through.

My robot design intends to model an extremely persuasive beggar. This project can be used as part of a study of people's reactions to beggars, a study on ways to avoid beggars, or to collect statistical data on the type of people who are willing to donate money to strangers. The following sections describe the system components, platform, actuators, sensors, robot behaviors, and experimental procedures and results of my robot.

Integrated System

Figure 1 shows a block diagram of the complete integrated system. BeggarBot is controlled by a 68HC11 mounted on a MTJPRO11 board. The sensory system is composed of 2 IR detectors, 4 bump switches, a photointerrupter, and a pyroelectric sensor. The actuating system consists of 2 servos modified for continuous rotation to drive the robot, 1 standard servo used as a sweeper, and 1 standard servo used for pulling. 6 NiCd AA batteries will power the overall system.

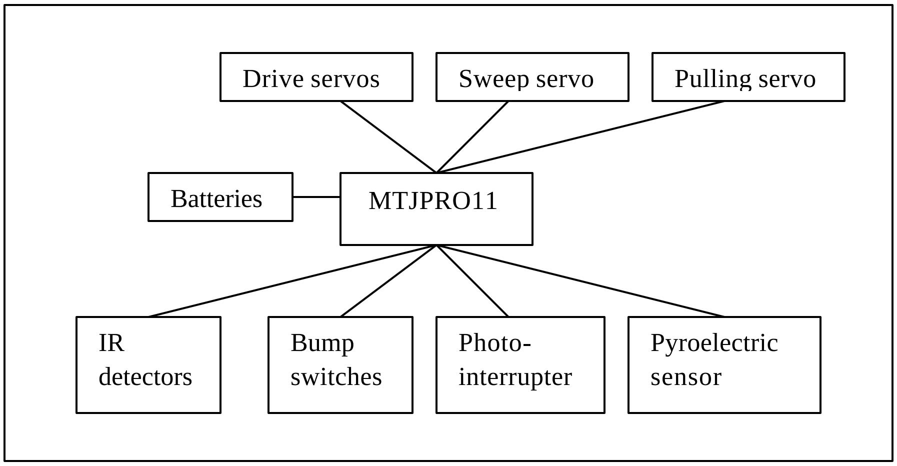


Fig.1 Complete Integrated System

Mobile Platform

BeggarBot's platform resembles the basic shape of a TJPro. Another level was added on top in order to support the coin repository and the shooting device. The platform was designed to house all the sensors and actuators in a compact and aesthetically pleasing manner. Figures 2 and 3 show the structural parts of the body. The round platforms were redesigned in the second board to support bump detection along with larger holes for mounting screws.

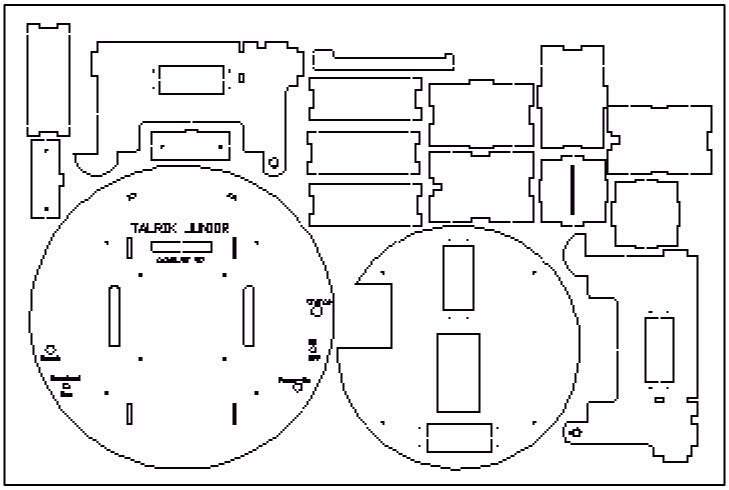


Fig.2 First Board

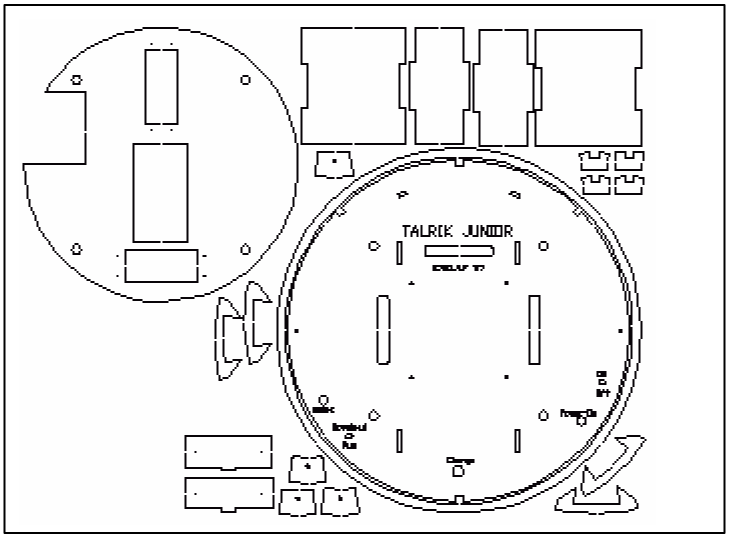


Fig.3 Second Board

Actuation

1) Drive motors

Purpose: The drive motors will control the movement of the robot.

Theory: A DC motor is the obvious solution to this problem. However, a servo can be hacked to act like a DC gear-head motor. The advantage is that these hacked servos can be driven directly from the TJPro board without motor drivers.

Application: Two hacked servos with wheels attached to the horns will act as the drive motors for BeggarBot.

2) Pyroelectric sensor sweep

Purpose: The sweeper will control the movement of the pyroelectric sensor.

Theory: Servos have limited rotation for precision angular control. This property can be used to position a device at different angles.

Application: The pyroelectric sensor will be mounted on the servo horn so the sweep angle can be controlled by the servo.

3) Trigger puller

Purpose: The trigger puller will be used to depress the trigger on the shooting device.

Theory: The movement of a servo is limited to 180 degrees of rotation. By sending proper PWM signals, the servo arm can be moved from one extreme to the other.

Application: A servo will be connected to shooting device in order to control the trigger action.

Sensors

1) IR emitter/detector (Part #: GP2D12)

Purpose: This sensor detects the amount of IR at a specific frequency.

Theory: This part is composed of both the IR emitter and detector. The emitter outputs a modulated IR beam that is tuned to the detector's sensitivity. The detector outputs an analog voltage that is proportional to the intensity it sees.

Application: The IR sensors will be mounted at the front of BeggarBot. One will be placed at the left front, the other at the right front. While executing collision avoidance, once either detector obtains a reading above a maximum threshold, the robot will turn to avoid the object.

Vendor: Mekatronix
316 NW 17th Street, Suite A
Gainesville, FL 32603
352-376-7373

2) Bump switch

Purpose: Bump switches are basic push-button switches that let you know if the button is pressed or not.

Theory: While the button is depressed, the terminals are short-circuited. When the button is not depressed, the terminals are open.

Application: The bump switches will be placed at the front and back of the robot's platform as a backup to the IR detectors. Along with aiding in collision avoidance, the bump switches will also be used to activate the robot upon reset.

3) Pyroelectric sensor (Part #: R3-PYRO1)

Purpose: Pyroelectric sensors detect heat, outputting a voltage proportional to the amount that it detects. This particular sensor is tuned to the human body's infrared emissivity.

Theory: The detector is sensitive to all optical radiation. The detector window limits the energy reaching the detector to about 7 - 16 micrometers. Humans emit a variety of wavelengths with the wavelength of maximum energy around 9 micrometers. These values are independent of race, but can be affected by clothing and inhibited circulation due to smoking. This detector uses lithium tantalite for its active substrate element since it is sensitive in the 8 - 14 micrometer range. Each side of the substrate is doped with an electrode. When infrared energy hits the substrate, a charge is developed between the electrodes, which is then amplified and used as the output. This pyroelectric sensor uses a dual element design that provides better performance than a single element design.

Application: This sensor will be mounted on a servo at the front of the robot. The servo will sweep left-right and right-left in order to detect both moving and stationary humans.

Vendor: www.acroname.com

4) Photointerrupter (Part #: Sharp 1A34LC)

Purpose: A photointerrupter indicates if an object passes through it's detection slot.

Theory: The photointerrupter uses an IR emitter mounted on one side of the detection slot and an IR detector mounted on the other side of the detection slot. The output of this device tells you when the IR beam has been broken.

Application: The photointerrupter will be mounted on the inside of the coin repository immediately underneath the coin slot. This will allow the robot to know if a coin has been placed in the collector.

Vendor: spare printer parts from the TA

Behaviors

1) Obstacle avoidance:

BeggarBot uses 2 IR emitter/detector pairs along with 4 bump switches to implement obstacle avoidance. The left and right IR detectors tell how close an object is in front of the robot. This allows BeggarBot to adjust its course to avoid colliding with the object. If the IR detectors do not detect an object, the bump switches allow the robot to stop moving whenever there is contact. The response of the robot to IR and bump data will depend on the specific action being performed at the time.

2) Human search:

BeggarBot will search for a human using a pyroelectric sensor that sweeps back and forth, detecting body heat. Once a human is detected, or what it thinks is a human, BeggarBot will position itself in front of the person.

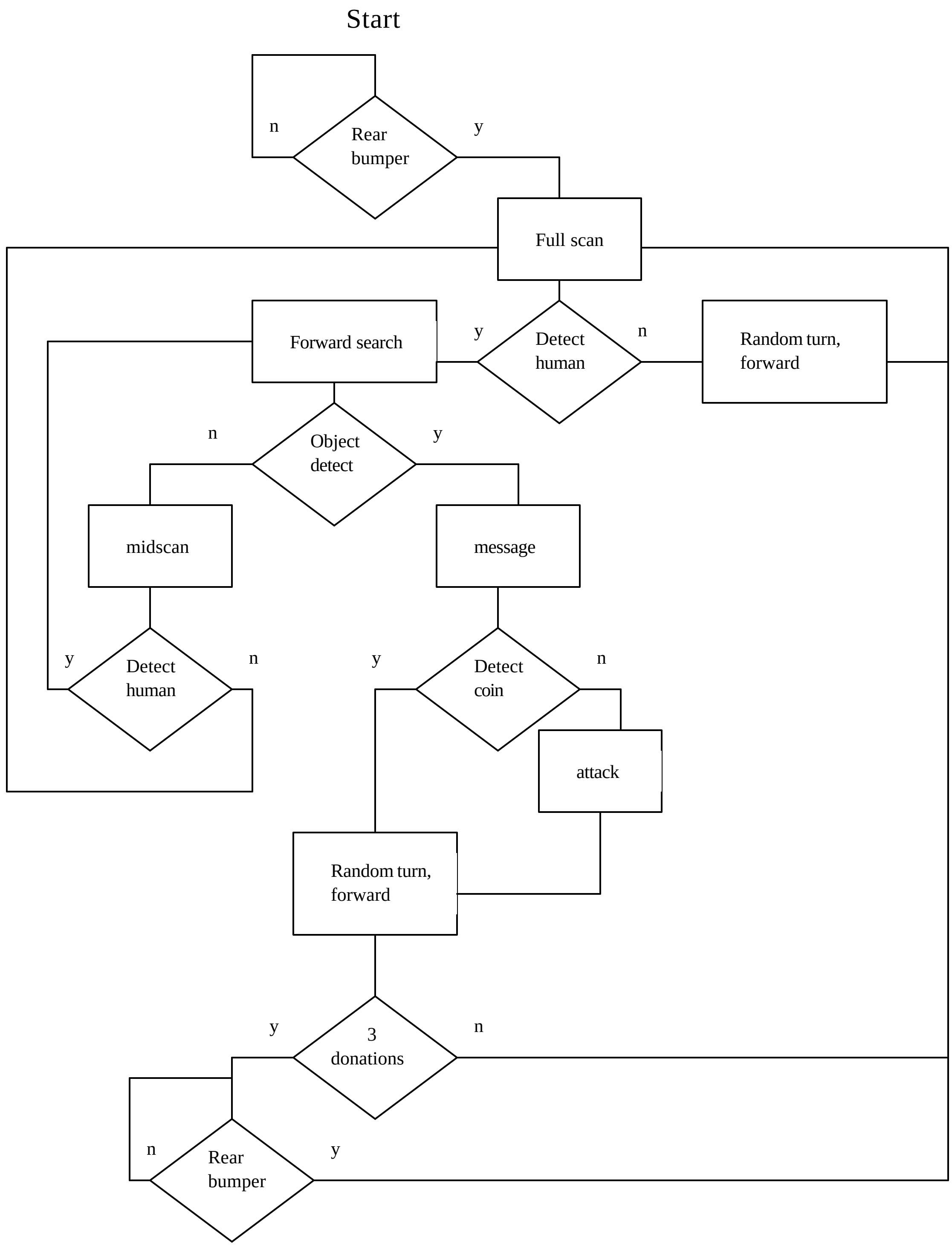
3) Leave if satisfied:

Once the robot is positioned in front of the human, it will expect a small donation to be placed in the coin repository. If BeggarBot detects that a donation has been made, it will turn around and start searching for another person. Once BeggarBot has obtained enough money for the day, it stops all movement and waits for its owner.

4) Shoot if not satisfied:

If BeggarBot detects that a donation has not been made, it will get very angry and shoot at the person standing in front of it.

Operational Flow Chart



Experimental Layout and Results

1) IR sensitivity

Objective: Obtain the characteristics of each IR detector.

Procedure: I used an A/D port to read and print out the value of the IR detector as I held a sheet of white paper in front of the device at different distances.

Results: The maximum reading for both detectors occurred at 4 inches. The left value was 144 and the right value was 145. Figure 4 shows the results over a range of 2 feet.

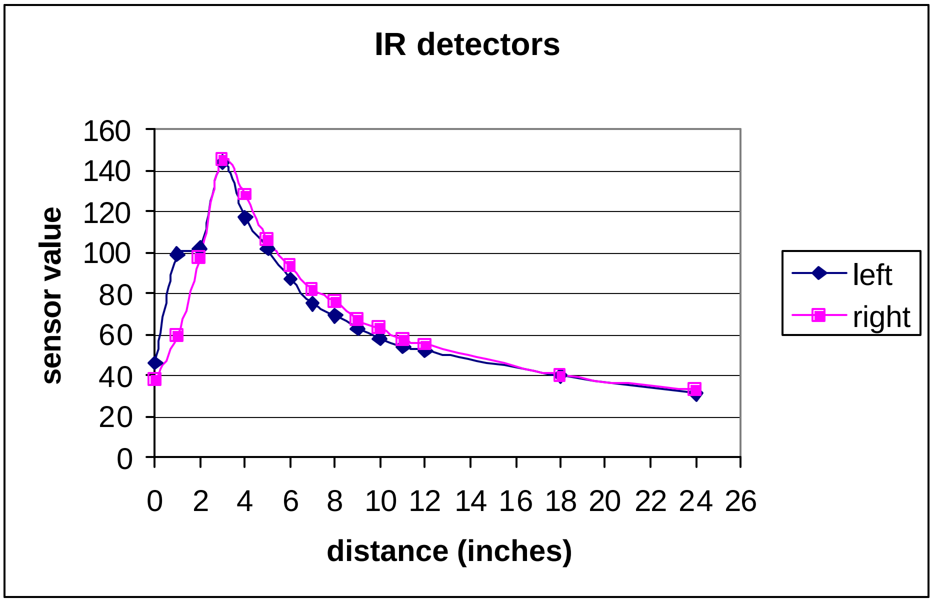


Fig.4 IR characteristics

2) Pyroelectric sensitivity

Objective: Obtain the characteristics of the pyroelectric sensor.

Procedure: I used an A/D port to read and print out the value of the sensor output. While the sensor was sweeping, I positioned myself at various distances in front of the robot and recorded the reading.

Results: Figure 5 shows a graph of an average of the data that I obtained. I am setting the threshold for the sensor at 140 since the smallest human detect reading I obtained during the individual trials was 141. The output hovers between 125 - 135 when there is nothing to detect. Clothing, as opposed to bare skin, seems to lower the reading of the output. I noticed the sensor is fooled by televisions and computer monitors. The sensor probably does not filter out the radiation they emit properly. I also noted that the output occasionally gets stuck at its high or low voltage level. This is eventually corrected as the sensor continues to sweep.

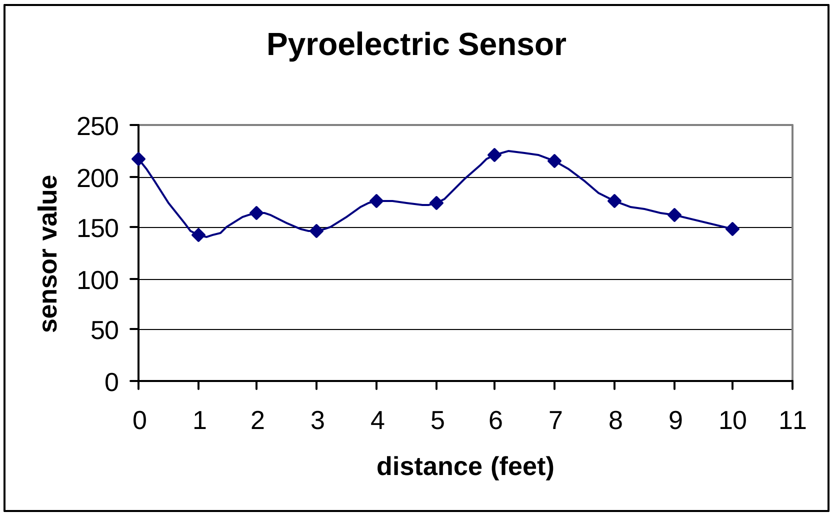


Fig.5 Pyroelectric characteristics

Conclusion

I am satisfied with the final state of BeggarBot. All actuators and sensors work well enough for the applications of my robot. BeggarBot can run for over 15 minutes with the 6 NiCd AA battery pack. If I were to start over with my design, I would have a more creative body, one that does not resemble a modified TJ Pro. One other modification would be to have the ability to differentiate between different types of coins and act based upon those observations. This would require a redesign of the coin collector and perhaps a different sensor such as a load cell. Besides these two considerations, all other specifications of my design would stay the same.

Documentation

[1] Keith L. Doty, Gainesville, FL,
TJPro Assembly Manual; Mekatronix 1999.

Appendices

Final Robot Code

```
/*************************** Includes ********************************/

#include <tjpbase.h>
#include <stdio.h>

/*************************** Constants ********************************/

#define GUNSERVO 0
#define PYROSERVO       2

#define COIN   analog(5)
#define PYROPORT analog(6)

#define LED_ON      *(unsigned char *)(0x7000) = 0x80
#define LED_OFF     *(unsigned char *)(0x7000) = 0x00

#define AVOID_THRESHOLD      90

/*********************** Function Prototypes******************************/

void stopmotors();
void turnright(int time);
void turnleft(int time);
void forward(int time);
void reverse(int time);
void randomturn();
void randomforward();
int forward_search(int time);
int objectdetect();
int init_scan();
int mid_scan();
void clearpyro();
int coin_detect();
void attack();
void message();

/**************************** Main *********************************/

int full;

void main() {
  init_motortjp();
  init_servotjp();
  init_analog( );
  init_serial();
  init_clocktjp();

  full = 0;

  LED_ON;
  while(BUMPER<120);
  LED_OFF;
  while(1) {
B1: if (init_scan()) {
B2:           servo(PYROSERVO, 1500);
      if (forward_search(2500)) {
                // BEG
                message();
                if (!(coin_detect())) {
                  reverse(1000);
                  attack();
                }
                randomturn();
                randomforward();
                if (full >= 3) {
                  START;
                  full = 0;
                }
                goto B1;
              } else {
        if (mid_scan()) {
          goto B2;
        } else {
          goto B1;
        }
      }
    } else {
      randomturn();
              randomforward();
    }
  }

}

/************************* Function Definitions************************/

void stopmotors() {
  motorp(LEFT_MOTOR, 0);
  motorp(RIGHT_MOTOR, 0);
}

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

void turnright(int time) {
  motorp(LEFT_MOTOR, 100);
  motorp(RIGHT_MOTOR, -100);
  wait(time);
  stopmotors();
}

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

void turnleft(int time) {
  motorp(LEFT_MOTOR, -100);
  motorp(RIGHT_MOTOR, 100);
  wait(time);
  stopmotors();
}

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

void forward(int time) {
  int i;
  motorp(LEFT_MOTOR, 100);
  motorp(RIGHT_MOTOR, 100);
  for (i=time; i>=0; i=i-50) {
    if (objectdetect()) {
      randomturn();
      randomforward();
      break;
    } else {
      wait(50);
    }
  }
  stopmotors();
}

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

void reverse(int time) {
  motorp(LEFT_MOTOR, -100);
  motorp(RIGHT_MOTOR, -100);
  wait(time);
  stopmotors();
}

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

void randomturn() {

  int i;
  unsigned rand;

  rand = TCNT;

  i=(rand % 1024);
  if (i <= 250) i = 250;

  if (rand & 0x0001) {
    turnleft(i);
  }
  else {
    turnright(i);
  }

}

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

void randomforward() {

  int i;
  i=(TCNT % 5000);
  if(i>100) forward(i); else forward(1000);

}

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

int forward_search(int time) {
  int i;
  int rval = 0;
  motorp(LEFT_MOTOR, 100);
  motorp(RIGHT_MOTOR, 100);
  for (i=time; i>=0; i=i-50) {
    if (objectdetect()) {
      rval = 1;
      break;
    } else {
     wait(50);
    }
  }
  stopmotors();
  return rval;
}

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

int objectdetect() {

  if ((RIGHT_IR > AVOID_THRESHOLD) || (LEFT_IR > AVOID_THRESHOLD)) {
    return 1;
  } else if (FRONT_BUMP) {
    return 1;
  } else {
    return 0;
  }

}

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

int init_scan() {

  int i,high,pyro,rval;
  rval = 0;
  high = 0;
  clearpyro();
  motorp(LEFT_MOTOR, -100);
  motorp(RIGHT_MOTOR, 100);
  for (i=2000; i>=0; i=i-2) {
    //printf("{%d}\n", analog(6));
    pyro = PYROPORT;
    if (pyro > high) high = pyro;
    wait(2);
  }
  stopmotors();
  wait(1000);

if (high > 141) {
  LED_ON;
  motorp(LEFT_MOTOR, -100);
  motorp(RIGHT_MOTOR, 100);
  for (i=2000; i>=0; i=i-2) {
    //printf("{%d}\n", analog(6));
    pyro = PYROPORT;
    if (pyro > high-10 && pyro > 141) {
      rval = 1;
      break;
    }
    wait(2);
  }
  stopmotors();
  LED_OFF;
}

  return rval;
}

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

int mid_scan() {

  int i,high,pyro,rval;
  rval = 0;
  high = 0;
  turnright(90);
  clearpyro();
  motorp(LEFT_MOTOR, -100);
  motorp(RIGHT_MOTOR, 100);
  for (i=180; i>=0; i=i-2) {
    //printf("{%d}\n", analog(6));
    pyro = PYROPORT;
    if (pyro > high) high = pyro;
    wait(2);
  }
  stopmotors();
  wait(1000);

if (high > 141) {
  LED_ON;
  turnright(180);
  clearpyro();
  motorp(LEFT_MOTOR, -100);
  motorp(RIGHT_MOTOR, 100);
  for (i=180; i>=0; i=i-2) {
    //printf("{%d}\n", analog(6));
    pyro = PYROPORT;
    if (pyro > high-10 && pyro > 141) {
      rval = 1;
      break;
    }
    wait(2);
  }
  stopmotors();
  LED_OFF;
}

  return rval;
}

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

void clearpyro() {

  servo(PYROSERVO, 2650);
  wait(1000);
  servo(PYROSERVO, 2950);
  wait(100);
  servo(PYROSERVO, 2650);
  wait(100);
  servo(PYROSERVO, 2950);
  wait(100);
  servo(PYROSERVO, 2800);
  wait(500);

}

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

int coin_detect() {

  int i;
  for (i=100; i>=0; i=i-10) {
    COIN;
    wait(10);
  }
  LED_ON;
  for (i=5000; i>=0; i=i-10) {
    if (COIN > 20) {
      full++;
      wait(1000);
      LED_OFF;
      return 1;
    }
    wait(10);
  }
  LED_OFF;
  return 0;

}

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

void attack() {

  servo(PYROSERVO, 2800);
  wait(500);
  // pull trigger
  servo(GUNSERVO,3200);
  wait(500);
  servo(GUNSERVO,0);
  wait(1000);
}

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

void message() {

  SET_BIT(PORTA,0x20);
  wait(250); // message length
  CLEAR_BIT(PORTA,0x20);
  wait(5000);

}

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

Analog test code

```
/************************** Includes ********************************/
#include <tjpbase.h>
#include <stdio.h>
/********************** End of includes *****************************/

void main(void)
/**************************** Main *********************************/
{
  int port,val;

/* VT100 clear screen */
  char c1, clear[]= "\x1b\x5B\x32\x4A\x04";
```

```
/* VT100 position cursor at (x,y) = (3,12) command is "\x1b[3;12H"*/
  char place[]= "\x1b[1;1H";   /*Home*/


  init_clocktjp();
  init_analog();

  printf("%s", clear);
  printf("%s", place);

  printf("Enter analog port number (0-7):   ");
  port = read_int();

while(1)
{
        val = analog(port);
        //printf("analog({%d}) value: {%d}\n",port,val);
        printf("%d\t",val);
        wait(10);

}

}
/**************************** End of Main ******************************/
```

Bumper test code

```
#include <analog.h>
#include <motortjp.h>
#include <clocktjp.h>
#include <isrdecl.h>
#include <vectors.h>
#include <stdio.h>

#define BUMPER              analog(0)

void main(void) {

        int i,value;
        init_analog();
        while(BUMPER<120);
        while(1) {
                value = BUMPER;
                printf("bump value: {%d}\n",value);
                for (i=0; i<30000; i++);
                for (i=0; i<30000; i++);
                for (i=0; i<30000; i++);
        }

}
```

IR test code

```
#include <analog.h>
#include <motortjp.h>
#include <clocktjp.h>
#include <isrdecl.h>
#include <vectors.h>
#include <stdio.h>

#define BUMPER              analog(0)
#define LEFT_IR       analog(2)
#define RIGHT_IR      analog(3)


void main(void) {

        int i,irdr,irdl;
        init_analog();
        while(BUMPER<120);
        while(1) {
                irdr = RIGHT_IR;
                irdl = LEFT_IR;
                printf("Right IR value: {%d}\n",irdr);
                printf("Left  IR va lue: {%d}\n",irdl);
                for (i=0; i<30000; i++);
                for (i=0; i<30000; i++);
                for (i=0; i<30000; i++);
        }

}
```

Motor test code

```
/**************************** Includes ***********************************/
#include <tjpbase.h>
#include <stdio.h>
/************************ End of includes *******************************/

void main(void)
/****************************** Main ***********************************/
{
  int i,lspeed, rspeed, servo_index;
  unsigned int PW;

/* VT100 clear screen */
  char c1, clear[]= "\x1b\x5B\x32\x4A\x04";

/* VT100 position cursor at (x,y) = (3,12) command is "\x1b[3;12H"*/
  char place[]= "\x1b[1;1H";   /*Home*/

//  init_servotjp();
  init_motortjp();
  init_serial();
  init_clocktjp();

while(1)
{

  printf("%s", clear);
  printf("%s", place);

  printf("Enter percent of left motor speed, a number between -100 and 100:   ");
  lspeed = read_int();

  printf("Enter percent of right motor speed, a number between -100 and 100:  ");
  rspeed = read_int();

  motorp(RIGHT_MOTOR, rspeed);
  motorp(LEFT_MOTOR, lspeed);

  printf("Type any character to input new data.");
  c1 = get_char();
}

}
/**************************** End of Main ******************************/
```

Pyro test code

```
/**************************** Includes **********************************/
#include <tjpbase.h>
#include <stdio.h>
/************************ End of includes *******************************/

void main(void)
/****************************** Main ***********************************/
{
  int i,j,data,high,pyro;
  init_servotjp();
  init_analog();
  init_serial();
  init_clocktjp();

while(BUMPER<20);

while(1)
{

  printf("\n");
  servo(2,1800);
  wait(500);
  for (i=2200; i<=3800; i+=400) {
    high = 0;
    servo(2, i);
    for (j=70; j>=0; j=j-2) {
      pyro = analog(6);
      if (pyro > high) high = pyro;
      wait(2);
    }
    printf("{%d}\n", high);
  }
  //data = read_int();

}


}
/**************************** End of Main ******************************/
```

Servo test code

```
/**************************** Includes **********************************/
#include <tjpbase.h>
#include <stdio.h>
/************************ End of includes *******************************/

void main(void)
/****************************** Main ***********************************/
{
  int i,lspeed, rspeed, servo_index;
  unsigned int PW;

/* VT100 clear screen */
```

```
  char c1, clear[]= "\x1b\x5B\x32\x4A\x04";

/* VT100 position cursor at (x,y) = (3,12) command is "\x1b[3;12H"*/
  char place[]= "\x1b[1;1H";   /*Home*/

  init_servotjp();
//  init_motortjp();
  init_serial();
  init_clocktjp();

while(1)
{

  printf("%s", clear);
  printf("%s", place);

  printf("\nEnter Servo Index: 0 to 2 (PA4-PA6)  ");
  servo_index = read_int();

  printf("Enter Servo Pulse Width: 1000<PW<4000 or PW=0:   ");
  PW = read_int();

  servo(servo_index, PW);

  printf("Type any character to input new data.");
  c1 = get_char();
}

}
/**************************** End of Main *****************************/
```