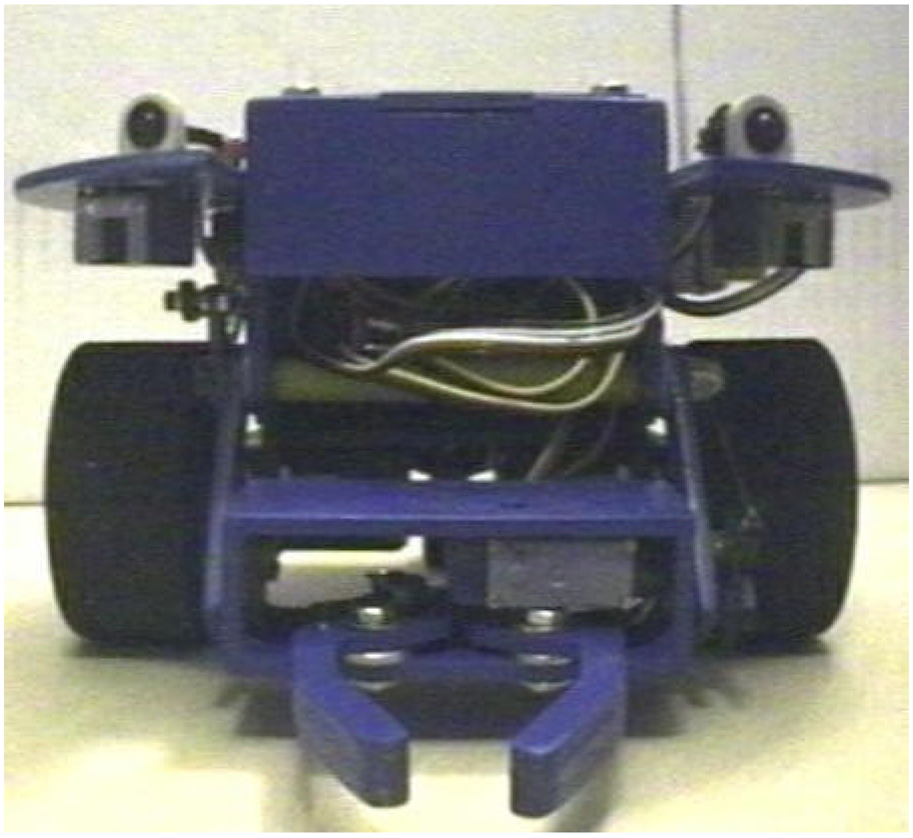
SMURF

TJLRWG-1

An IMDL Spring 2000 Project



Boris Yaw

EEL 5666

Professor: Dr. A.A. Arroyo

TABLE OF CONTENTS

ABSTRACT

Smurf is an autonomous small object manipulating robot The design goal behind Smurf was to build a manipulating robot that would fit in the same physical dimensions as a Talrik Junior (TJ). With this in mind Smurf was designed to fit into a cylinder 7 inches in diameter and 4 inches tall with a retractable arm that extends about three inches outside of this cylinder. Other design goals included low cost, and a finished or consumer ready look. Smurf is designed to pick up an object, and is capable of carrying this object in its gripper to another location. Ideal objects would fit in a 2 inch cube and weigh less than 8 ounces. Smurf uses 5 servos(1 sub micro servo for the grippers and four standard servos, two for propulsion and two for arm motion) Smurf is controlled using a motorola 6811 microcontroller and uses bump switches, IR, CDS cells and a microphone to navigate around a room, find small objects and move them.

EXECUTIVE SUMMARY

Smurf's original name was TJLRWG-1, or Talrik Junior(TJ) Like Robot With Gripper, but after it was painted all blue it was called Smurf by other IMDL students since it was also one of the smallest robots in the lab.

The objective was to build a small, low cost, manipulating robot that would be similar in size to a Talrik Junior. The two robots share the same two servo propulsion system, and Smurf is constructed out of aircraft ply just like the TJ. Smurf also has its main navigation IR system mounted in a very similar position to the TJ's. Smurf has a circular shape when viewed from above, another shared feature. This symmetrical, circular shape comes in handy when navigating in small spaces, since the robot essentially has no corners to worry about, and is especially handy for this type of robot that can rotate about its central axis.

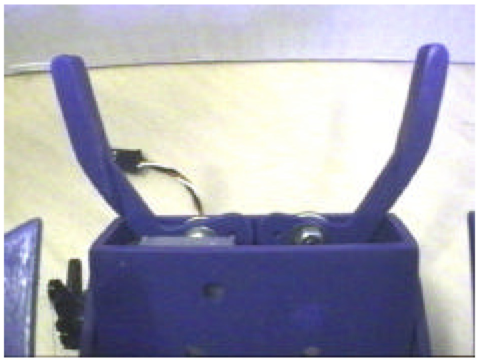


Fig. 1, A view of Smurf's gripper from above.

Smurf's biggest difference from a TJ is the gripper located at the front of the robot. This gripper can be opened and closed and can be moved up or down. The gripper can also be rotated.

For basic object avoidance, TJ code can be used without any modification. Smurf is not just a modified TJ, and though it was designed to be similar, it had to be designed with some major differences. Instead of just having two servos, it was necessary to mount 4 servos within Smurf's body. Then room was needed for the retractable arm, which would also fit inside the volume of a normal TJ. Smaller diameter tires were used, but these are of a different style than those on the TJ. Smurf's tires resemble racing slicks, and are 1 inch wide, with all of that 1 inch touching the floor. These were used to provide better traction

Smurf has a sensor suite of three forward looking IR emitter/detector pairs, three downward pointing CDS cells under the gripper mechanism, five bump switches around its perimeter and on the gripper and a microphone.

Smurf has a piezo-electric buzzer which can be used to communicate with its users. Examples of such communication might be, two beeps to indicate that he is waiting for a user's action, or a long irritating buzz to indicate that an object is blocking his path.

Smurf's purpose is to be a small, compact object manipulator. It was designed for indoor use only, and can possibly be used to move items around on a table or counter.

INTRODUCTION

Two areas consumed the most of my time in this class, one was designing up with the platform, and the second was programming the robot.

It was assumed that with a well designed platform, the necessary code to operate it would be simple, and the robot would have a wider variety of uses. I also wanted to make the robot as low cost as possible. The 1/8” aircraft plywood used in the robot was included in the lab fee, The four standard servos from Mekatronix cost $40, and the wheels (two main wheels and one castor) cost about $15. The six cell battery pack and microcontroller seemed to be standard items in the IMDL robots. It seemed like I would have to get those regardless of the type of robot I made. Other items that were of significant out of pocket expenses included three IR E/D pairs ($12) a sub micro servo($25) and a can of blue spray paint ($3.50). I also wanted to make my robot reproduceable, so I designed the body using AutoCAD, and cut out the parts using the T-Tech machine in the lab.

Programming the behaviors seemed to take forever. I used ICCTJP, and soon learned of its limitations. Thankfully there were lots of previous examples of TJP code, and these came in very handy.

MOBILE PLATFORM

A simple box like shape constructed from interlocking pieces of aircraft ply (cut on the MIL T-Tech machine) was made to house the electronics. To keep things simple it was decided that a rolling, indoor platform would be used. I wanted to use as many ideas from previous IMDL classes as possible, that way I

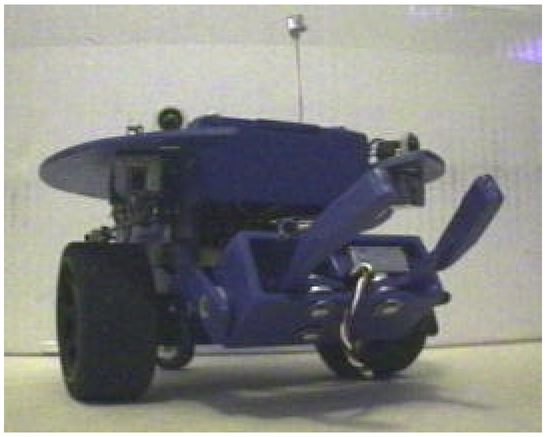
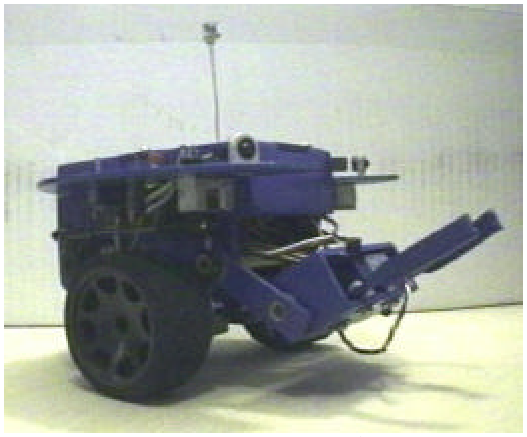



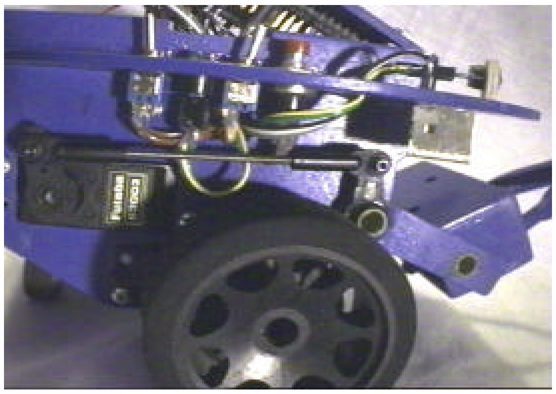
Fig. 2 & 3, Smurf viewed from the front. The microphone sensor can be seen at the top of the images. This was an initial stage where only two IR detector/emitter pairs are mounted.

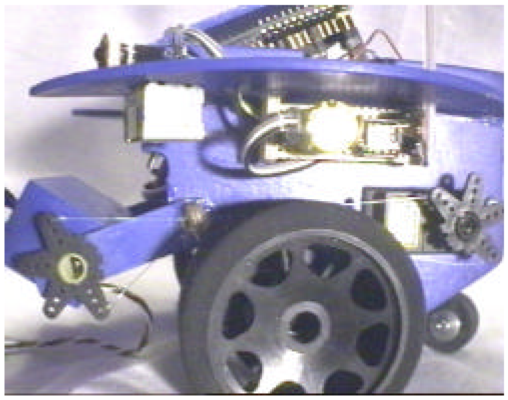
could ensure that I would have a robot that worked and the job of putting it together would be easier. A two degree of freedom arm was mounted to this box, and a gripper was installed on the end of it. Circular "fenders" were then added to the simple box like design. These were used to mount IR sensors, a small control panel, an amplifier and a "bump ring" similar to that used on a TJ. The entire robot was cut from 1/8" aircraft ply. The arm was made movable by using carbon fiber tubing and nylon washers in the joints.

ACTUATION

Basic navigation is accomplished in a manner identical to that of a Talrik Junior. The robot is steered by independently powered wheels, one on each side. The wheels are powered by hacked standard servos that now function as small geared motors. The wheels are mounted on one of the centerlines of the robot, which as in the TJ, ensures that the robot can rotate about its own central axis if necessary. A small castor was designed to support the rear portion of the robot. The center of mass was located between the drive wheels and the castor to ensure that the robot didn't tip over when moving. When the robot backs up suddenly slight tipping still occurs.

One servo on the right side of the robot moves the arm up and down and another servo changes the angle of the gripper. A tiny sub micro servo mounted in the gripper mechanism opens and closes the gripper.





Figs. 4 & 5 The photo on the left shows the servo mechanism that raises and lowers the gripper. The photo on the right shows the servo mechanism that tilts the gripper. This push-pull arrangement proved to be faulty and was replaced in the final design. The amplifier for the microphone is also visible in the photo on the right.

SENSORS

Smurf's sensor suite was selected for low cost, availability of code and simplicity of implementation. The primary goal of the sensors are large object avoidance (navigation) and small object location.

BUMP SWITCHES

A bump ring, or set of three bump switches encircled by a ring of plywood is used from bump sensing. This is primarily used in navigation, when the IR emitters fail to detect objects because of their size, shape or color. This ring is also used to communicate with the robot. Touching the rear bump sensor may be used to start the robot.

IR SENSOR

Three IR emitter/detector pairs are used. Two are used in object avoidance and one used for small object location.   Basically the IR emitter directs a 40 kHz beam or IR away from the robot, and a modified Sharp IR detector looks for a reflection. Based on the intensity of this reflection the hacked IR detector produces an analog signal. A threshold can then be set in software so that these IR emitter/detector pairs can be used to detect the proximity of objects within a certain range. IR is primarily a close range detector which seems to work best at distances under 12 inches, although objects several feet away have been located in the lab. These sensors have been used extensively by IMDL robots and are well documented on the IMDL web page. Data gathered for my sensor report showed that the maximum values for my IR was about 128 (on the 0-255 scale for analog input), and this occurred from about 8 inches away from the detector. I used a threshold of between 100 and 110 (object about 1 foot away) in my software so that the robot would be able to react fast enough.

<u>CDS CELLS</u>

Three CDS or Cadmium Sulfide cells are used on the underside of the gripper for line following. Using three of these cells, mounted perpendicular to smurfs direction of forward motion, smurf can be used to follow a black line on a light background. CDS cells are very hard to use since they are so sensitive to light. Even small changes such as the angle of overhead light can cause changes in their sensed data. Other factors include angle or presence of sunlight and type of light, florescent or other. CDS cells are best used when shielded from these external light sources, and a small local light source such as a bright LED is used with the CDS cell so that the light source is always kept constant. Values for the CDS cells ranged from 0-245, 0 being total darkness, and 245 being florescent light.

<u>MICROPHONE</u>

Smurf has one microphone, mounted on a plastic tube, high above the rest of the robot. Smurf listens for loud noises and reacts to them. In the program included with this report (The Demo Day Program), smurf does a 180 degree turn whenever he hears a loud noise. The microphone/amplifier was made from parts obtained from Radio Shack, Electronics Plus, and the IMDL lab. The circuit for the amplifier was found in Jones & Flynn's "Mobile Robots: Inspiration to Implementation". I had a problem with my amplifier after I built it to Jones 7 Flynn's specs. I kept getting a sine wave for analog input, which was no good. I had to put a small resistor in between the capacitor and pin one of the LM386. Ambient noise gave values of around 160, I set my noise threshold to be about 175, and Smurf would then react to handclaps.

BEHAVIORS

Smurf is capable of using standard TJ pro software (avoidtj.c) for random motion and collision avoidance behavior.

Smurf is also capable of simple line following behavior using the CDS cells mounted under the gripper.

For demo day, smurf was programmed with a special type of hunting behavior. The robot was programmed to follow a line, which widened to large dark spots at the ends. The CDS cells used to follow the line detected the dark spots and the microcontroller is programmed to recognize these as "endpoints". Smurf would traverse this line looking for items to pick up on the endpoints of the line. The single IR detector mounted on the gripper is used to detect whether or not an item is there to pick up. Smurf is programmed to beep when he arrives at an endpoint. This lets the user know when he has arrived. If there is nothing to pick up, he does a 180 degree turn and goes to check on the other end of the line. The microphone senses loud noises and these cause Smurf to turn around. The user could use this if they needed Smurf to return quickly. If an object blocks its path, the robot stops and turns on a buzzer, bringing the obstruction to the users attention. When the IR sensors no longer detect an obstruction, the robot continues on its path.

Smurf also has a random or demo behavior, which is a modification of the standard avoidtj software. Besides just doing avoid behavior, smurf stops at random intervals and looks for something to pick up. His search pattern is very predictable, so if you see him searching you can usually put something down in his expected path for him to find. When it has something, it picks the object up and wanders around with it looking for a dark spot on the floor to put the object down on.

DISCUSSION

I didn't realize how long it would take to write the software for this robot. Perhaps its because one has to modify the code, download it to the robot then run the robot through several test cases. Then to make things worse you have to decide wether the bugs are due to hardware or dead batteries.

I used either full speed or none at all when moving the robot because there was some sort of interference between the motortjp code and the servotjp code, so that when running servomotors and servos together, the servomotors can't be controlled very precisely.

The gripper didn't seem to work too well and I think I need to redesign one that touches the object to be picked up with more surface area. The way mine worked its like trying to pick up objects with one's finger tips as opposed to the entire hand area.

The sub micro servo broke right before demo day and the gripper did not work as designed. The gripper mechanism was modified to use one of the standard servos, the one usually used to change the angle of the gripper, instead of a sub micro servo, to open and close the gripper.

CONCLUSION

Smurf works reasonably well considering the time constraints of the class. If I had more time I would spend a little more time designing the gripper itself. I would change the shape a little so that more of the gripper's surface area is in contact with the object being picked up. I would also try to design a gripper that would use a standard servo to open and close instead of a micro servo. The battery pack area needs to be redesigned also to allow someone to change the batteries without having to open up the robot as much as is currently necessary. David Grindlinger, fellow IMDL student, suggested putting metal plates somewhere on the grippers, so that the robot could recharge itself whenever it sensed that its batteries were dying, this seems like a good idea and may be done in the next version of this robot.

Appendix A – Part List

Item	**Source**
1 Six Cell battery pack 6 individual NiCads + Case w/9 volt connector	IMDL Lab
4 Standard Servos	Mekatronix
2 1/10 scale RC car racing slicks	Gator Hobby
1 RC aircraft tail wheel – used with some music wire, brass tubing, & washers to make castor	Gator Hobby
1 sheet of 1’ x 2’ 1/8” aircraft ply	IMDL Lab
3 CDS cells	IMDL Lab
3 IR emitter LEDs & 3 Sharp IR detectors	Mekatronix
1 Condenser Mic Element	Radio Shack (part #270-090c)
1 LM386 amp chip	Electronics Plus
Misc. resistors, wires, connectors, switches & capacitors	IMDL Lab
MTJPRO11 32K microcontroller board	Mekatronix
1 12vdc 10mA Piezo Buzzer	Radio Shack (part #273-059)
Motorola 6811 chip	IMDL Lab
4 Du-bro 2-56 swivel/ball links	Gator Hobby
1 FMA Direct s-80 sub micro servo	Gator Hobby
Nylon Washers	Lowes Hardware
Carbon Fiber tubing	Personal stock(this looks so cool, must have a use on the robot)
Rubber lining for inside of gripper	Wal-Mart(from Ping-Pong paddles)
2-56 screws, nuts & washers	IMDL Lab
2-56 threaded rod	Gator Hobby

Appendix B – Source Code for Demo Day Program

```
/*****************************************************************************
 *                    IMDL Spring 2000                           *
 * Title          hunt.c                                *
 * Programmer      Boris Yaw                                *
 * Date           April 28, 2000                           *
 * Version 1                                            *
 *                                                 *
 * Description                                           *
 *    A simple "hunting behavior" for the tj like robot with gripper    *
 *    also known as Smurf.  Smurf follows a line and looks for objects  *
 *    at the end points of this line.  If an object is in the path,     *
 *    Smurf stops and buzzes.  If there is nothing to pick up, Smurf    *
 *    does a 180 degree turn and tries the other end of the line.       *
 *    Smurf also responds to loud noises.                          *
 *                                                 *
 ***************************************************************************/



/**************************** Includes *************************************/
#include <tjpbase.h>
#include <hc11.h>
#include <mil.h>
/************************* End of Includes ********************************/


/**************************** Global Variables *****************************/
int speedr, speedl;

/**************************** End global variables ***************************/

/*************************** Prototypes ********************************/
void setdigport(void);
void stopbot(void);
void buzz_on(void);
void buzz_off(void);
void turnaround(void);
void forward(void);
void backup(void);
void turn_left(void);
void turn_right(void);
void rotate_right(void);

/************************ End of Prototypes ****************************/


/****************************** Main ***********************************/
void main(void)
{
  int i, ready, full, empty, grip_up, grip_down;
  int irdl, irdr;

  init_analog();
```

```
init_motortjp();
init_clocktjp();
init_servotjp();
setdigport();

IRE_ON;     /* turn on IR emitters */

empty=1;
full=0;
grip_down=0;
grip_up=1;
ready=0;

while(1)
{
                        while(empty)
      {
        servo(0, 1000);
        servo(1, 3000);

        irdr = RIGHT_IR;
        irdl = LEFT_IR;

        if ((irdl > 110) &&
            (irdr > 110))
        {
           stopbot();   /* stop robot */
           buzz_on();   /* turn buzzer on */
        }

   /*********************************************************************/
        else
        {
           buzz_off();

        if (analog(7) > 180) /* check the mic level */
        {
             stopbot();     /* if there is a noise, turn around */
             turnaround();
             forward();
        }

        if ((analog(6) < analog(4)) &&  /* compare the right CDS cell to the center */
            (ready < 1))
        {
            turn_right();              /* if its over a line turn right */
        }


        if ((analog(1) < analog(4))  &&  /* do the same for the left cell */
            (ready < 1))
        {
            turn_left();
        }
        if ((analog(4) < analog(1)) &&
            (analog(4) < analog(6)) &&
            (ready < 1))
        {
            forward();
        }

/****************************************************************/
        if ((analog(4) < 150) &&
            (analog(1) < 150) &&
            (analog(6) < 150) &&
            (ready < 1))
        {
             stopbot();
             buzz_on();
             wait(250);
             buzz_off();
             wait(250);
             buzz_on();
             wait(250);
             buzz_off();
             wait(2000);

            /*********************************************/
            /* dart forward if something is there to pick up */

            if (analog(5) > 100)  /* check the center CDS cell */
            {
              forward();
              wait(500);

              stopbot();
              wait(500);

              for(i=3000; i>999; i-=1) /*close gripper*/
              {
                servo(1, i);
                wait(2);
              }

              for(i=1000; i<2801; i+=1) /* raise arms */
              {
                servo(0, i);  /*SERVOZ, PA4*/
                wait(2);     /*Wait 2 milliseconds between commands*/
              }
              empty=0;
              full=1;
              grip_up=1;
              ready=1;

              backup();
              wait(500);

              stopbot();
            for(i=2800; i>1499; i-=1) /* lower arms */
            {
               servo(0, i);  /*SERVOZ, PA4*/
               wait(2);     /*Wait 2 milliseconds between commands*/
            }

          }
         /*********************************************/

             turnaround();
             wait(850);

             forward();
         }
/**************************************************************/
         } /***** end of else ******/

     }  /* end while(empty) */



  /**************************************************************/
  /**************************************************************/
     while(full)
     {


       if (analog(7) > 180)
       {
           stopbot();
           turnaround();
           wait(825);

           forward();
       }

           if (analog(6) < analog(4))
           {
             turn_right();
           }

           if (analog(1) < analog(4))
           {
             turn_left();
           }

           if ((analog(4) < analog(1)) &&
              (analog(4) < analog(6)))
           {
             forward();
           }


/**************************************************************/
           if ((analog(4) < 150) &&
              (analog(1) < 150) &&
              (analog(6) < 150))
            {
              stopbot();

              if ((grip_up) &&
                  (full))
              {

                for(i=1500; i>999; i-=1) /* lower arms */
                {
                   servo(0, i);  /*SERVOZ, PA4*/
                   wait(2);     /*Wait 2 milliseconds between commands*/
                }

                for(i=1000; i<3001; i+=1) /* open gripper */
                {
                   servo(1, i);  /*SERVOZ, PA4*/
                   wait(2);     /*Wait 2 milliseconds between commands*/
                }
                backup();
                wait(500);

                rotate_right();
                wait(850);

                stopbot();

                ready=0;
                grip_down=1;
                grip_up=0;
                full=0;
                empty=1;

              }
            }


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




       }  /* end while(full) */
   }       /*  end while(1) */

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


/************************ Function setdigport *************************/
/* Obtained from Cynthia Manya ****************************************/
void setdigport(void)
{
 SET_BIT(DDRD, 0x08);                          /* set PortD bits 3 & 4 as dig outputs */
 SET_BIT(DDRD, 0x10);
 CLEAR_BIT(PORTD, 0x08);                    /* turn sound off (set bit 3 to 0 volts) */
 CLEAR_BIT(PORTD, 0x10);                    /* turn sound off (set bit 4 to 0 volts) */
}
/*********************** End Function setdigport ***********************/

/************************* Function stopbot *****************************/
void stopbot(void)
{
  speedr = 0;
  speedl = 0;
  motorp(RIGHT_MOTOR, speedr);
  motorp(LEFT_MOTOR, speedl);
}
/*********************** End Function stopbot ***********************/

/************************* Function buzz_on*************************/
void buzz_on(void)
{
  SET_BIT(PORTD, 0x08);
  SET_BIT(PORTD, 0x10);
}

/*************************End Function buzz_on *************************/

/************************* Function buzz_off*************************/
void buzz_off(void)
{
  CLEAR_BIT(PORTD, 0x08);
  CLEAR_BIT(PORTD, 0x10);
}

/*************************End Function buzz_off *************************/

/************************* Function turnaround*************************/
void turnaround(void)
{
     speedr = -MAX_SPEED;
     speedl = MAX_SPEED;
     motorp(RIGHT_MOTOR, speedr);
     motorp(LEFT_MOTOR, speedl);
     wait(825);
}

/************************* Function turnaround*************************/

/************************* Function forward*************************/
void forward(void)
{
     speedr = MAX_SPEED;
     speedl = MAX_SPEED;
     motorp(RIGHT_MOTOR, speedr);
     motorp(LEFT_MOTOR, speedl);
}

/************************* End function forward*************************/
```

```
/************************** Function backup**************************/
void backup(void)
{
     speedr = -MAX_SPEED;
     speedl = -MAX_SPEED;
     motorp(RIGHT_MOTOR, speedr);
     motorp(LEFT_MOTOR, speedl);
}

/************************** End function backup**************************/

/************************** Function turn_left**************************/
void turn_left(void)
{
    speedr = MAX_SPEED;
    speedl = 0;
    motorp(RIGHT_MOTOR, speedr);
    motorp(LEFT_MOTOR, speedl);
}
/************************** End function turn_left**************************/

/************************** Function turn_right**************************/
void turn_right(void)
{
    speedr = 0;
    speedl = MAX_SPEED;
    motorp(RIGHT_MOTOR, speedr);
    motorp(LEFT_MOTOR, speedl);
}
/************************** End function turn_right**************************/

/************************** Function rotate_right**************************/
void rotate_right(void)
{
    speedr = -MAX_SPEED;
    speedl = MAX_SPEED;
    motorp(RIGHT_MOTOR, speedr);
    motorp(LEFT_MOTOR, speedl);
}
/************************** End function rotate_right**************************/
```