Team Five: Erin Keith & Qandeel Sajid
Lab 2
Date: 9/5/12

Introduction

This lab is an extension of lab one in which the team designed and built the HandyBug robot. In this lab the students were expected to use Interactive C to implement and test the following five programs: Beeper, Motor, Sensor, Obstacle, and Multi-tasking. This report discusses the implementation of these programs, any problems that were encountered during the lab, and the results from testing the programs on the robot.

Description

An example of the HandyBug robot is shown in Figure 1. As shown, the robot was built using LEGOs. The instructions on how to build the HandyBug were found in book *Robotic Explorations* (2001) by Fred G. Matrin. The robot includes two touch sensor cables that are attached to the front of the HandyBug and are triggered by the bumper. There are also two motor cables that are connected to the back wheels. On top of the HandyBug is the Handy Board that is used to control the robot.

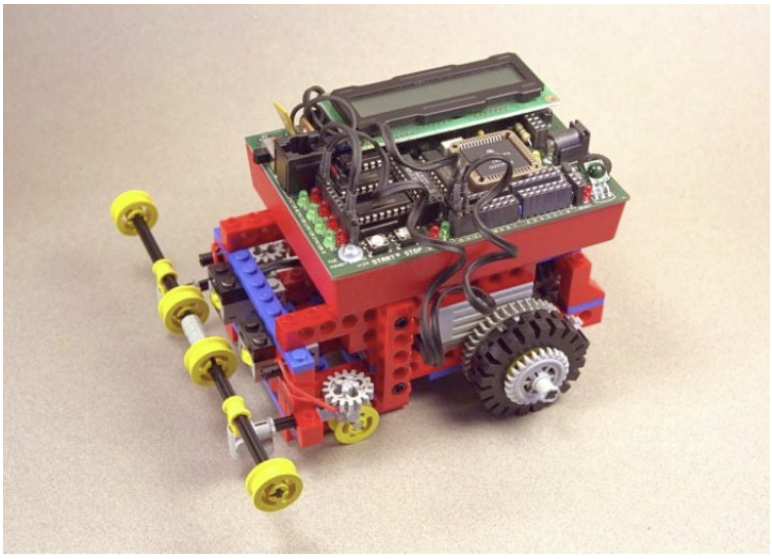

Figure 1 - Demonstrates a sample HandyBug robot.

In the Beeper program, the robot beeps for 3 seconds with 500Hz frequency, then stays silent for one second, beeps again for 3 seconds with 100Hz frequency, then stays silent for one second and finally beeps for another two seconds with 1000Hz frequency. The program is shown in the Appendix. To implement this program the students just used the functions tone, and sleep.

The Motor program requires the robot to turn left for three seconds, reverse for two second, beep then turn off for two seconds and then turn on with full power for three second, beep, and turn off. For implementation, the students used toe motor function to control which motor turned on and at what power. Function fd(), and bk() were used to move the robot forwards and backwards.

In the Sensor program, when one of the robot's sensors is touched, it needs to turn around until that sensor is touched again. In the sensor program the global variable run is used to control when the robot is on or off. When it is one, the robot moved forward and checks for collisions. If either of its sensors detects a collision, the robot will turn print out which sensor was triggered.

Then the waits one tenth of a second before turning off its right motor (to rotate left) until that same sensor is triggered again. Then run is set to zero to terminate the loop running the robot.

In the Obstacle Avoidance program, when one of the robot's sensors is triggered, the robot will go backwards and rotate, and then continue going forwards again. For this program, it needs to be able to turn on with the press of the start button and off when the stop button is pressed. This is why the start_button() and stop_button() functions where used. The robot waits in the while loop for the start button, once it is pressed it goes into a while loop waiting for the stop button. In the second loop, the robot moves forward, and check if either of its sensors have been triggered. If so, it will stop, move backwards for half a second, robot left for about 90 degrees, and then go back to moving forwards until the sensors are triggered again.

The Multi-tasking program is the same at the Obstacle Avoidance program except that is multi-threaded. For this program, to make the code more clear and legible, the team makes use of global variables and functions. The numbers correlating to the motors and sensors are now global variable so they are easier manage. There are two functions: move_forward and check_sensors. The move_forward function moves forwards when there is no obstacle, else it stops and rotates to avoid the obstacle. Whether or not the robot has collided with an obstacle is determined by the value of the global variable OBSTACLE. The check_sensors sets the value of OBSTACLE to one when either of the sensors is triggered. To use the multi-threading the team needed to use the function start_process provided by Interactive C. move_forward and check_sensors are started after the start button is pressed and killed after the stop button is pressed.

Difficulties Encountered

We had a minor problem during the Sensor program where hitting an obstacle once would cause the robot's sensors to be triggered twice consecutively causing the robot to stop rather than rotate. To solve this problem we just set the robot to sleep for one tenth of a second after the first collision check.

Discussion of Unsolved Problems

There were no unsolved problems in this lab.

Results

When tested, all of the programs worked as instructed.

Conclusion

In conclusion, the students learned how to implement and test minor Interactive C programs on the HandyBug during this lab.

Appendix

Beeper Program

```
{
tone(500.0, 3.0);
sleep(1.0);
tone(100.0, 3.0);
```

```
sleep(1.0);
tone(1000.0, 2.0);
}
```

Motor Program

```
int main()
{
     motor(0, 50);
     sleep(3.0);
     beep();

     motor(0, -50);
     sleep(2.0);
     beep();

     off(0);
     sleep(2.0);
     fd(0);
     sleep(3.0);
     beep();
     off(0);
}
```

Sensor Program

```
int main()
{
  int run = 1;

  while( run )
  {
    fd(0);
    fd(3);

    if( digital(11))
    {
      printf("1st Sensor");
      sleep(0.1);
      off(0);

      while( !digital(11));

      printf("2nd Sensor");
      run = 0;
      off(3);
    }



    else if( digital(10))
```

```
    {
      printf("1st Sensor");
      sleep(0.1);
      off(3);


      while( !digital(10));

      printf("2nd Sensor");
      run = 0;
      off(0);
    }
  }

}
```

Obstacle Avoidance Program

```
int main()
{
     while( !start_button());

     while( !stop_button())
     {
         fd(0);
         fd(3);

        if( digital(11) || digital(10))
         {
             off(0);
             off(3);

             bk(0);
             bk(3);
             sleep(.5);

             fd(0);
             sleep(1.7);

              off(0);
              off(3);
         }
     }

    off(0);
    off(3);
 }
```

Multi-Tasking Program

```
int RT_MTR = 0;
int LT_MTR = 3;
```

```
int RT_SNSR = 10;
int LT_SNSR = 11;

int OBSTACLE = 0;

int main()
{
   int pid1, pid2;
   while( !start_button());

   pid1 = start_process( move_forward());
   pid2 = start_process( check_sensors());

   while(!stop_button());

   kill_process(pid1);
   kill_process(pid2);

   off(RT_MTR);
   off(LT_MTR);
}

void move_forward()
{
   while(1)
     {
      fd(RT_MTR);
      fd(LT_MTR);

      if( OBSTACLE )
        {
          off(RT_MTR);
          off(LT_MTR);

          bk(RT_MTR);
          bk(LT_MTR);
          sleep(.5);

          fd(RT_MTR);
          sleep(1.7);

          off(RT_MTR);
          off(LT_MTR);

          OBSTACLE = 0;
       }
```

```
   }
}

void check_sensors()
{
   while(1)
     {
       if( digital(LT_SNSR) || digital(RT_SNSR))
         OBSTACLE = 1;
     }
}
```