Jian Huang

Intelligence Machine Design Laboratory

Tennis Ball Fetcher

Summer, 1998

Table of Contents

ABSTRACT

I designed an autonomous robot named “ Tennis Ball Fetcher.” Its purpose is to collect tennis ball and send it back to home base. The robot will autonomously navigate through a tennis ball field and collect the ball.

The body was made of one-quarter inch birch plywood. It used a break-beacon to detect the ball and a gate mechanism to hold the ball. The main computing power is the HC6811 microcontroller and the ME11 expansion board. I used the Sonar proximity sensor to avoid obstacles, IR sensors to detect the ball, and sonar sensors as home base beacon. I also used bump sensors.

EXECUTIVE SUMMARY

I designed an autonomous robot named “Tennis ball Fetcher” to collect the tennis ball in the tennis court and send the ball back to home base. The body of the robot is made of one-quarter inch birch plywood.

The brain of the robot is the HC6811 and ME11 expansion board. I used the 40kHZ sonar sensors to navigate through the field and the 24 kHZ sonar sensor as the home base beacon. IR sensor is used to detect ball, and the bump sensors are also used. I used a gate mechanism to hold the ball.

INTRODUCTION

I designed an autonomous robot named “Tennis ball fetcher” to collect ball and send them back to a home base. The programming language is the Interactive C. Interactive C is developed by MIT, and it supports multi-tasking. The Sonar proximity sensor is used for collision avoidance and the home base beacon.

I used the IR break beacon to detect the ball and the gate mechanism to hold the ball. The body of the robot is made of one-quarter inch plywood, two motors and a caster. The brain of the robot is the HC6811 and ME11 expansion board.

SENSORS

IR SENSORS

The IR sensors consist of a LED IR emitter and “sharp” receiver. The receiver is hacked from digital receiver to analog receiver. The details of hacking can be found at www.mekatronix.com. The IR sensor is used as a break-beacon to detect the ball. The design is as follows: If no ball is between the IR emitter and receiver, then the reading is 129 from the Analog-to-Digital port. If there is a ball between the beacon, the reading of the analog-to-digital port is 92.

SONAR SENSORS

I used two 40 kHZ sonar sensors as the proximity sensors. The transmitter and most of the receiver circuits can be found at the final paper of Michael Apodaca of Spring 98 at www.mil.ufl.edu/imdlf.html. The 40 kHZ signal is directly from the ME11 expansion board. The change I made at the receiver is that instead of feeding the output of the MAX266

Active filter into a digital comparator, I feed it into a diode in series with a low pass filter. The diode is used as a rectifier since the output of the active filter is a sine wave. The low pass filter consists of a resistor and a capacitor in series. The corner frequency of the low pass filter is below 40 kHZ so that I get a DC level signal out of the low pass filter. The value I used for the resistor is 47 ohms and the capacitor is 0.1 microfarads. I turned the sonar transmitter on and off for about 2 milliseconds. I did this to avoid the standing wave of the sound. The experiment I conducted showed that for continuously turning the sonar sensor on will not give consistent collision avoidance data because of the standing wave.

BUMP SENSORS

The bump sensor is a spring with a wire in the middle of the spring. When the spring touches the wire, there is a contact between the wire and the spring and the voltage at the contact is fed into the analog port. I used four bump sensors, and the sensor is just a voltage divider circuit when the switch is closed, otherwise the voltage is at 5 volts.

HOME BASE SENSOR

The home base sensor is 24 kHZ sonar sensor. I used the MAX038 to generate the 24 kHZ signal and the MAX266 chip to pick up the signal on the receiver. In the MAX266 chip, I used the three second-order bandpass filters cascaded in series.

ACTUATION

I used a servo to control the gate. When the robot is wandering, I lift up the gate so that the ball can go into the gate and be detected by the break beacon.

BEHAVIORS

COLLISION AVOIDANCE

I used the two sonar sensors to do the collision avoidance. The algorithm is simple. When the left sonar detects something but not the right one, the robot turn right. When the right sonar sensor detects something but not the left one, the robot turn left. When both sensors detect something, the robot goes backwards. Otherwise, the robot goes straight.

BALL DETECTION

When a ball is detected between the gate, that is the IR beacon is broken by the ball, the robot closes the gate.

HOMEBASE BEACON

After detecting the ball, the robot will spin to find the maximum beacon and go towards the beacon. When the robot does not go very straight, it will correct itself by spin again and go straight. The details of the code are described by the code in the appendix.

CONCLUSION

Through this class, I designed a robot and met my most design goals. Building a robot needs a lot of engineering skills and discipline. Thank Dr. Arroyo and Dr. Schwartz for providing the lab facility outside the lab hours and TA’s extra help outside their schedule.

APPENDIX

```
int LEFT=0;
int RIGHT=0;
int OLD_LEFT=0;
int OLD_RIGHT=0;
int THRESHOLD=4;
int LEFT_MOTOR=0;
int RIGHT_MOTOR=1;
float NORM=60.0;
float STOP=0.0;
int BOTTOM=0;
int TOP=0;
int FLAG=0;
int OLD_BOTTOM=0;
int PID;
int PID2;
int MAX_HOME=0;
int SONAR_RE[1500];
int MAX_SONAR=211;
int SONAR=0;

void stopp()
{
 motor(LEFT_MOTOR, 0.0);
 motor(RIGHT_MOTOR, 0.0);
}

void go_right()
{
 motor(LEFT_MOTOR, NORM);
 motor(RIGHT_MOTOR, STOP);
}

void go_left()
{
 motor(RIGHT_MOTOR, NORM);
 motor(LEFT_MOTOR, STOP);
}

void go_forward()
{
 motor(LEFT_MOTOR, NORM);
 motor(RIGHT_MOTOR, NORM);
}

void go_forward_little()
{
 motor(LEFT_MOTOR, 20.0);
 motor(RIGHT_MOTOR, 20.0);
}
```

```
void go_back()
{
 motor(LEFT_MOTOR, -1.0* NORM/2.0);
 motor(RIGHT_MOTOR, -1.0* NORM/2.0);
}

void find_dir()
{
  while(1){
    motor(LEFT_MOTOR, 30.0);
    motor(RIGHT_MOTOR, -30.0);
    if (analog(0)>(MAX_HOME-5) && analog(0)<(MAX_HOME+5)){
     stopp();
     break; }
  }
}

void spin()
{
  int i;
  for (i=0; i<1500; i++){
    motor(LEFT_MOTOR, 50.0);
    motor(RIGHT_MOTOR, -50.0);
    SONAR_RE[i]=analog(0);
  }
}

void find_max()
{
  int i;
  for(i=0; i<1500; i++){
     if (SONAR_RE[i]>MAX_HOME)
          MAX_HOME=SONAR_RE[i];
  }
}


void irscan()
{
 while(1){

  poke(0x7000, 0xf1);
  poke(0x7000, 0xf0);
  LEFT=analog(5);
  RIGHT=analog(4);

 }
}

void waits(int milli_seconds)
{
 long timer_a;
 timer_a=mseconds()+(long)milli_seconds;
```

```
    defer();
  }
}

void avoid_obstacle()
{
 while(1){
  if (LEFT - OLD_LEFT>THRESHOLD && RIGHT-
OLD_RIGHT<=THRESHOLD){
    go_right();
    waits(1000);
  }
  else if (RIGHT-OLD_RIGHT > THRESHOLD && LEFT-OLD_LEFT<=
THRESHOLD){
    go_left();
    waits(1000);
  }
  else if (RIGHT-OLD_RIGHT>THRESHOLD && LEFT-
OLD_LEFT>THRESHOLD){
    go_back();
    waits(1000);
  }
  else {
      go_forward();
      BOTTOM=analog(6);
      if ( (OLD_BOTTOM-BOTTOM) >30  && FLAG==0 ){
            FLAG=1;
         stopp();
         waits(2000);
            go_forward();
            waits(1500);
              servo_deg(32.5);
            waits(2000);
            servo_off();
            defer();
    }
  }

 }

}

void find_h(){
while (1){
  if (FLAG==1){
    stopp();
    waits(1000);
    go_forward();
    waits(1500);
    servo_deg(32.5);
    waits(2000);
    servo_off();
    kill_process(PID);
```

```
    spin();
    find_max();
    find_dir();
    SONAR=analog(0);
    while(1)  {
      go_forward();
      waits(1500);
      if (SONAR>analog(0)){
     stopp();
        spin();
        find_max();
        find_dir();
        SONAR=analog(0);}
      else if(analog(0)>207 && analog(0) < MAX_SONAR){
        stopp();
        waits(2000);
        break;}
    SONAR=analog(0);
    }
    FLAG=0;
    servo_on();
    servo_deg(120.0);
    waits(2000);
    go_forward();
    waits(250);
    go_back();
    waits(2500);
    go_right();
    waits(500);
    PID=start_process(avoid_obstacle() );
    PID2=start_process(irscan());
  }
  else defer();
}

}


void main()
{
    LEFT=0;
    RIGHT=0;
    OLD_LEFT=0;
    OLD_RIGHT=0;
    THRESHOLD=4;
    LEFT_MOTOR=0;
    RIGHT_MOTOR=1;
    NORM=60.0;
    STOP=0.0;
    BOTTOM=0;
    TOP=0;
    FLAG=0;
    OLD_BOTTOM=0;
```

```
    MAX_SONAR=211;
    SONAR=0;
  servo_off();
  servo_on();
  servo_deg(120.0);
  stopp();
  waits(3000);
  OLD_LEFT=148;
  OLD_RIGHT=170;
  OLD_BOTTOM=129;
  PID2=start_process(irscan());
  PID=start_process(avoid_obstacle() );
  start_process(find_h());
/*  start_process(find_ball() ); */

}
```