Lab 04
Jessica Smith and Luke Fraser
Team 06

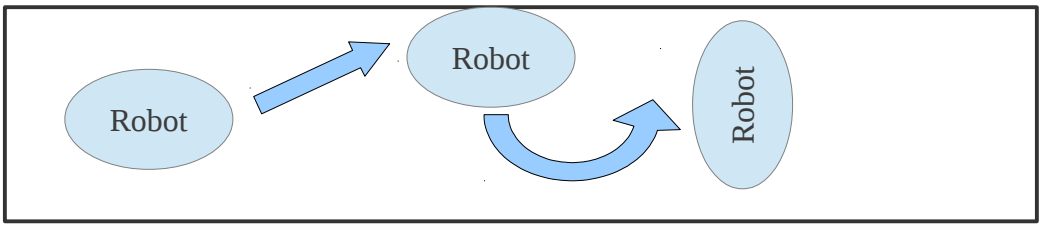
Introduction

The goal of this lab contest was to design a robot that would gather “food” for one minute and then find its way back to the starting point. The food items were RFID cards, both small and large, and colored pieces of paper. The sensors we used for this project were a color sensor and the RFID sensor also used in the last lab. To detect walls, we used sonar sensors. The team that gathered the most food and got back home the fastest would be the winner.

Description

Our robot was designed with the two sonar sensors lying almost flat on the ground, at about a 50 degree angle outwards in order to detect the walls more efficiently. If the sonar sensors were angled too sharply or too shallowly, they wouldn't correctly detect the walls and would get stuck in corners. With the angle we chose as well as the distances we used in our program, getting stuck wasn't easy to do. However, our sensors for the wall-following portion of our seek-home routine did not perform as well as other robots. The sensor wouldn't detect the closeness of the wall as accurately, likely due to the angle we chose, and would eventually end up sliding against the wall. This strategy worked well enough until the robot hit a bump in the wall which caught the sensor and caused it to turn into the wall. However, because we were so good at detecting being stuck in corners, it would correct this behavior and would not become stuck. It just wasn't a very fast way to search for home. Figure 1 shows the general pattern of behavior of the search for home.

Figure 1: Search for home issues, the pattern of running into the wall



The process we used to search for food was to go as fast as we could in a zig-zagging pattern to cover as much ground as possible. We used the approach to make small turns as we went because we noticed that the sensors often missed their targets by a small amount, or hit a corner and didn't cover the right amount of the target to actually detect it. To try to account for this, the robot made a sweeping motion as it moved around the map. The goal was to give the sensors more time to be detected. It worked less well than we hoped it would, because it still missed most of the food by a very small amount. Once the timer was up, it turned the motors on full power until it ran into a wall, and then it followed that wall until it found the home base. During the competition, we were lucky because our robot was facing perfectly towards home and found it almost instantly.

Problems

This competition had to main parts, and the first part was to find the food and the second part was to find home. We had a task which controlled most of the movement for the food-finding part of the competition. One problem we had was that we were not killing that task during the home-finding part of the competition. Since there were conflicting tasks, all the robot did was spin wildly in circles. Once we stopped that task, the robot behaved properly during the second half of the competition. Another problem we ran into was the angle of our sonar sensors was excellent for detecting if it was stuck, but not as good for following walls. We would repeatedly run along the wall until we turned to face the wall completely, do a reverse turn and repeat the process, as seen in Figure 1. It did get home eventually, but took a very long time. We never completely solved this issue. The robot as it runs now will follow the wall at a good distance for a short time, but will drift towards the wall until it is running along the wall and using it as a stability guide. Eventually if the wall has any deformities, the sensor would get stuck and it would cause the robot to do the turn into the wall issue that we saw before, but with much less frequency. This is in part due to our motors working at different levels of power. One is much stronger than the other, so we had to adjust for that in the code, which is why there are uneven values in our program. All we could do to compensate for this was guess at values, which was not very accurate.

Discussion

If given more time, we would adjust the robot's home-finding program. We would also probably do away with the sweeping motion of the robot, seeing as it didn't seem to make a very big difference. The home finding part of the program needs adjustment because of the problem discussed above and seen in Figure 1. We were able to adjust its sensitivity to the wall and the speed that it followed the wall so that most of the time it was just running up against the wall instead of making 90 degree turn after 90 degree turn, but it was still not as effective as other team's robots. The sweeping pattern was a good idea in theory, but the jerking motion involved with changing direction made the sensors move too much to pick up the food items.

Conclusions

This robot performed well enough in the competition to take third place, but that was mostly due to luck at our final search for home. We likely would not have placed had we had poor placement for the search for home portion of the competition. If we could do it again, we would change the functionality of the robot to perform better. However, with the strategies that we did implement, we did respectable.

Appendix

```
#include "RFIDlib.nxc"
// Food Finder Program

#define PORT1 IN_3
#define PORT2 IN_2
mutex moveMutex;

bool isturned = false;
bool findHome = false;
int THRESHOLD = 10;
int BLACK = 0;
int ColorCounter = 0;
```

```
int IDCounter = 0;
bool wall = false;
bool direction = false;

// function to do a turn to avoid a wall to the left
void turnaroundLeft(){
    OnFwd(OUT_B,-100);
    OnFwd(OUT_A,20);
    Wait(550);
    OnFwd(OUT_AB, 100);
    Wait(400);
}

// function to do a turn to avoid a wall to the left, on its route home
void turnaroundLeftHome(){
    OnFwd(OUT_B,-100);
    OnFwd(OUT_A,20);
    Wait(400);
    OnFwd(OUT_AB, 100);
    Wait(200);
}

// function to do a turn to avoid a wall to the right
void turnaroundRight(){
    OnFwd(OUT_B,20);
    OnFwd(OUT_A,-100);
    Wait(550);
    OnFwd(OUT_AB, 100);
    Wait(400);
}

// function to do a turn to avoid a wall to the right, on its route home
void turnaroundRightHome(){
    OnFwd(OUT_B,20);
    OnFwd(OUT_A,-100);
    Wait(400);
    OnFwd(OUT_AB, 100);
    Wait(200);
}

// In case of getting stuck or sensor malfunction, perform a random turn
void randomTurn(){
   int num = Random(30) + 50;
   OnFwd(OUT_B, -num);
   num = Random(30) + 50;
   OnFwd(OUT_A, num);
   Wait(400);
}
```

```
// Find the wall at the end of search period
void goWall(){
  int left = SensorUS(IN_4);
  int right = SensorUS(IN_1);

  while(1){
    int left = SensorUS(IN_4);
    int right = SensorUS(IN_1);
    if(right < 15){
      break;
    }
    if(left < 15){
      break;
    }
    OnFwd(OUT_B, 100);
    OnFwd(OUT_A, 95);
    Wait(200);
  }
  if(right < 15){
   direction = false; // follow right side
  }
  else{
    direction = true; // follow left side
  }
}

// Turn right to follow the wall in find home routine
void goRight(){
 while(1){
  int left = SensorUS(IN_4);
  int right = SensorUS(IN_1);
  int in = 20;
  int out = 25;

  if(right < in){ // turn slightly left
    OnFwd(OUT_B, 30);
    OnFwd(OUT_A, 70);
    Wait(100);
  }
  if(right > out){  // turn slightly right
    OnFwd(OUT_B, 70);
    OnFwd(OUT_A, 10);
    Wait(200);
  }
  if(right < in && left < in){ // turn around
    turnaroundLeftHome();
  }
  else{
    OnFwd(OUT_B, 80);
```

```
    OnFwd(OUT_A, 80);
    Wait(300);
   }
  }
}

// Turn left to follow the wall in find home routine
void goLeft(){
  while(1){
   int left = SensorUS(IN_4);
   int right = SensorUS(IN_1);
   int in = 20;
   int out = 25;

   if(right < in){ // turn slightly left
    OnFwd(OUT_A, 30);
    OnFwd(OUT_B, 70);
    Wait(100);
   }
   if(right > out){  // turn slightly right
    OnFwd(OUT_A, 70);
    OnFwd(OUT_B, 10);
    Wait(200);
   }
   if(right < in && left < in){ // turn around
    turnaroundRightHome();
   }
   else{
    OnFwd(OUT_A, 80);
    OnFwd(OUT_B, 80);
    Wait(300);
   }
  }
}

// This task moves the robot in a "straight" line during searching
task moveBothSideM(){
    while(findHome == false){
     Acquire(moveMutex);
     OnFwd(OUT_A, 75);
     OnFwd(OUT_B, 20);
     Wait(250);
     OnFwd(OUT_B, 70);
     OnFwd(OUT_A, 25);
     Wait(250);
     Release(moveMutex);
    }
}
```

```
// controls sensor input processing
task check_sensor(){
    while( findHome == false ){
      int left = SensorUS(IN_1);
      int right = SensorUS(IN_4);
       short gap = 10;
       if(left < gap)
       {
         Acquire(moveMutex);
         turnaroundLeft();
         Release(moveMutex);
       }
       if(right < gap){
         Acquire(moveMutex);
         turnaroundRight();
         Release(moveMutex);
       }
       if(right < gap && left < gap){
         Acquire(moveMutex);
         randomTurn();
         Release(moveMutex);
       }
    }
    while(findHome == true){
      int left = SensorUS(IN_4);
      int right = SensorUS(IN_1);

      // get to side
      Acquire(moveMutex);
      goWall();
      Release(moveMutex);

      if(direction == false){ // right
        Acquire(moveMutex);
        goRight();
        Release(moveMutex);
      }
      else{ // left
        Acquire(moveMutex);
        goLeft();
        Release(moveMutex);
      }
   }

}

// Continuously checks for food
task check_COLOR(){
    // send dummy command to wake up sensor
```

```
    RFIDDummy(PORT1);
    byte a[5];

    while(findHome == false){
       GetRFIDArray(PORT1,a,true);
       //ClearScreen();

       //NumOut(0,LCD_LINE1,10*(a[0]+a[1]+a[2]+a[3]+a[4]));
       if((a[0]+a[1]+a[2]+a[3]+a[4]) > 0){
          PlayTone(1200,300);
          IDCounter = IDCounter + 1;
          NumOut(1, LCD_LINE6, IDCounter);
       }
       else if(SensorHTColorNum(PORT2) < THRESHOLD){
          ClearScreen();
          ColorCounter = ColorCounter + 1;
          NumOut(1, LCD_LINE6, ColorCounter);
          PlayTone(1200,300);
       }
    }
    while(findHome == true){
      if(SensorHTColorNum(PORT2) == BLACK){
          ClearScreen();
          PlayTone(1200,300);
          Acquire(moveMutex);
          OnFwd(OUT_AB, 0);
          Wait(300);
          Release(moveMutex);
          TextOut(2, LCD_LINE6, "FOUND HOME" );
          StopAllTasks();
      }
    }
}

// timer task which calls the random function every 15 seconds
task timer(){
   Wait(15000);
   Acquire(moveMutex);
   randomTurn();
   Release(moveMutex);
   Wait(15000);
   Acquire(moveMutex);
   randomTurn();
   Release(moveMutex);
   Wait(15000);
   Acquire(moveMutex);
   randomTurn();
   Release(moveMutex);
   Wait(15000);
```

```
    Acquire(moveMutex);
    randomTurn();
    Release(moveMutex);
    findHome = true;
    ClearScreen();
    TextOut(1, LCD_LINE1, "FINDING HOME...");
    while(1){
      Wait(100);
    }
}


task main(){
SetSensorLowspeed(IN_1);
SetSensorLowspeed(IN_4);
SetSensorLowspeed(PORT1);
SetSensorLowspeed(PORT2);
SetHTColor2Mode(PORT2, HT_CMD_COLOR2_NEAR);
Precedes(timer, check_COLOR, moveBothSideM, check_sensor);
}
```