ARCH 879/ECE 893 - Architectural Robotics
Project 3 – Aging in Place
Team members – Paul Yanik and Tarek Mohktar

Abstract

"Loneliness is and always has been the central and inevitable experience of every man."
Thomas Wolfe

The system is based on an overhead 2-dof robot capable of prismatic motion along the length and width axes of a single room. A hanging armature equipped with infrared sensors can be used to pick up and deposit objects between any desired points in the room while avoiding obstacles. The robot uses 2 motor driven belts to pull the armature carriage along axis tracks.

ET uses IR sensors to detect the presence of obstacles in the robot's trajectory. When an obstacle is detected, the robot recalculates it's motion possibilities by assigning high cost to the obstructed directions. As shown here, when the observer's hand is placed in the direction of desired motion, the robot attempts to circumnavigate the obstacle by choosing a lower cost path to its goal. If no navigable trajectory is available, the robot holds its position.

A common problem faced by many people is that of loneliness. ET seeks to minimize the aspects of loneliness associated with aging by analyzing voice for emotional affect and by providing an appropriate automated response. The robot can detect four distinct emotional states: happiness, sadness, anger and fear. In each of these cases, the robot provides a personalized stimulus in the sensory areas of lighting color, ambient music, touch by bring a favorite piece of reading material, and taste by serving a favorite beverage.

Scenario1:

ET: Hi Lucille do you still remember me?
I am your ET friend!
L: Hi ET!
ET: How do you feel today?
L: I feel sad…
ET: Yes I see, please don't feel sad when I am with you, I will bring you a nice book to read and bring you a cup of orange juice.
L: Yes, please ☺….etc

When you feel happy, the sun will enter your room through the movement of the curtains and windows.. The light will be very bright and the speakers will start 'swan lake' music that you like, the Together will move to bring you Italian Folktales book from the shelf to your place without any need from you to control. It is like a real friend who understands you before you say to him what he has to do.
You are not alone you have Together who understands your mood and brings you things to have better mood, Together will close the doors of loneliness and opens the door to a new interactive life.

Hardware Used

- Arduino Duemilanove boards (2)
- Adafruit Motor/Stepper/Servo Shield for Arduino
- Parallax continuous rotation servo motor
- Pullup resistors
- Architectural (room) model
- Articulated arm lamp (as flower stem)
- Chipboard cutouts for outer coverings and decoration

Source Code (Robot Control)

```
// ----------------------------------------------------
// Team members:  Tarek Mohktar, Paul Yanik
//
// Description:
// This project implements a robot which seeks to assist
// a person who may be alone in their home by interpreting
// emotional states and tailoring robotic action to
// to tend to the persons needs in a positive manner.
// The robot is suspended overhead in a bedroom environment
// and is capable of moving over any point in the room.
// Hence, the robot manipulator can serve to automatically
// bring favorite items, control lighting, and music.
// This program serves to control the robot.
// ----------------------------------------------------
```

```
#include <AFMotor.h>
AF_Stepper yStepper(48, 1);

#include <ServoTimer1.h>
ServoTimer1 xServo;


// --------------------------------------------------------
// Pin Assignments and Variable Declarations
// --------------------------------------------------------
int voiceBit0 = 0;  // Voice affect selection bits:
int voiceBit1 = 0;  // 0 = happy, 1 = sad, 2 = angry, 3 = afraid.
// Presence sensor inputs
int presFront = 0;
int presBack = 0;
int presLeft = 0;
int presRight = 0;

// Variables
int motorRPM = 30;

int atGoal = 1;
int numWaypoints = 0;
int x = 0;
int y = 1;
int dist2goal = 0;

// Mood array row mnemonics
int happy = 0;
int sad = 1;
int angry = 2;
int afraid = 3;
int currMood = 0;
int lastMood = 0;

// Moods - each motion (row) consists of a number of waypoints (shown in column[0]),
// and (x,y) coordinates of each waypoint to the goal for a given mood.
int moods[4][5] = {
  {1,50,50}, // Happy
  {2,20,100,100,10}, // Sad
  {2,50,60,200,200}, // Angry
  {1,0,0}  // Default - go home
};

int currGoal[2] = {0,0};
int currPosn[2] = {0,0};
```

```
int nextMoves[2] = {0,0};

// ------------------------------------------------------
// IO Pin Setup
// ------------------------------------------------------
void setup()
{
  Serial.begin(9600); // set up Serial library at 9600 bps

  xServo.attach(9);

  // Setup analog input pins for reading
  for (int i=14; i<=19; i++) {
    pinMode(i, INPUT);
  }

  // Set the stepper speed
  // xStepper.setSpeed(motorRPM);
  yStepper.setSpeed(motorRPM);

  // xStepper.release();
  yStepper.release();
  delay(500);
}

// ------------------------------------------------------
// Loop for Runtime Operation
// ------------------------------------------------------
void loop()
{

  // Read voice affect inputs
  voiceBit0 = digitalRead(14);
  voiceBit1 = digitalRead(15);

  // Determine the goal positon based on input from the DAQ
  if ((voiceBit1 == LOW ) && (voiceBit0 == LOW )) {currMood = happy;}
  if ((voiceBit1 == LOW ) && (voiceBit0 == HIGH)) {currMood = sad;}
  if ((voiceBit1 == HIGH) && (voiceBit0 == LOW )) {currMood = angry;}
  if ((voiceBit1 == HIGH) && (voiceBit0 == HIGH)) {currMood = afraid;}

  Serial.println(currMood);

  // Wait for a change in mood.
  if (currMood != lastMood) {
```

```
// atGoal = 0;

// Find number of waypoints associated with the current mood.
numWaypoints = moods[currMood][0];

for (int k = 0; k < numWaypoints; k++) {

  atGoal = 0;
  currGoal[x] = moods[currMood][(2*k)+1];
  currGoal[y] = moods[currMood][(2*k)+2];

  while (atGoal == 0) {


    // Serial.println(atGoal);

     // Read obstacle sensors
     presFront = digitalRead(16);
     presBack  = digitalRead(17);
     presLeft  = digitalRead(18);
     presRight = digitalRead(19);


    dist2goal = abs(currGoal[x] - currPosn[x]) + abs(currGoal[y] - currPosn[y]);

    // Serial.println(dist2goal);

    // Determine attainment of goal by calculating the difference in
    // current position from the goal position.
    if (dist2goal < 3)
      {atGoal = 1;}
      else {atGoal = 0;};

    // Find the next motion for the robot based on current position
    // and any obstacles present.  Store in nextMoves array.
    findNextMove (currPosn[x], currPosn[y], currGoal[x], currGoal[y],
      presLeft, presRight, presFront, presBack, nextMoves);

    // Serial.println(currPosn[x]);
    // Serial.println(nextMoves[0]);

    currPosn[0] = currPosn[0] + nextMoves[0];
    currPosn[1] = currPosn[1] + nextMoves[1];

    // Turn the motors based on the nextMoves array contents.
    doubleStep (nextMoves[0], nextMoves[1]);
```

```
      } // while

      // Pause briefly between waypoints.
      delay(500);

    } // for
  } // if

  // Store value of last mood movement.
  // Only change positions if the mood changes.
  lastMood = currMood;

} // end loop

// ---------------------------------------------------
// Motor stepping function
// ---------------------------------------------------
void doubleStep (int stepsX, int stepsY) {

  int servoFwd = 95;
  int servoRev = 85;
  int servoStop = 90;
  int servoDelay = 200;

  // Go backward if the negative x/y direction if the
  // number of steps is negative.

  // A servo motor is used to move the manipulator
  // along the x axis.
  if (stepsX < 0) {
   xServo.write(servoRev);
   delay(servoDelay);
   xServo.write(servoStop);
  } else if (stepsX == 0) { // do nothing - no movement
  } else {
   xServo.write(servoFwd);
   delay(servoDelay);
   xServo.write(servoStop);
  }

  // A stepper motor is used to control the manipulator
  // along the y axis.
  if (stepsY < 0) {
    yStepper.step(-stepsY, FORWARD,  DOUBLE);
  } else if (stepsY == 0) { // do nothing - no movement
```

```
  } else {
    yStepper.step( stepsY, BACKWARD, DOUBLE);
  }

  // delay(50);

}

// ---------------------------------------------------------
// Directional control - find the number of steps to take
// and the direction of motion.
// ---------------------------------------------------------
void findNextMove
  (int currX, int currY, int goalX, int goalY, // Current positon and goal coordinates.
  int presF, int presR, int presL, int presB,  // IR sensor inputs.
  int *nextMoves) {                        // Array[2] of steps to take in x and y directions.

  int obstaclePenalty = 10000; // Obstacle cost penalty
  int numSteps = 1; // Multiply next move by numSteps to make movement faster.
  int stepX = 0; // Vector to move in the x direction.
  int stepY = 0; // Vector to move in the y direction.
  int score0, score1, score2,
      score3, score4, score5,
      score6, score7, myScore; // Distance costs for possible vector movements in each
direction.

  int F = 0;
  int R = 0;
  int L = 0;
  int B = 0;

  // Amplify distance to goal for each direction if an obstacle is sensed.
  F = obstaclePenalty * presF;
  R = obstaclePenalty * presR;
  L = obstaclePenalty * presL;
  B = obstaclePenalty * presB;

  // Determine the next position that is closest to the goal
  // by calculating distances from all surrounding positions.
  // Add penalties based on presence sensor activation (an obstacle
  // in any given direction (L,F,R,B).
  //
  // Positions:  7   6   5
  //             4  Curr 3
  //             2   1   0
```

```
  // One row below current position
  score0 = abs((goalX - (currX-1))) + abs(goalY - (currY-1)) + R + B;
  score1 = abs((goalX -  currX   )) + abs(goalY - (currY-1))  + B;
  score2 = abs((goalX - (currX+1))) + abs(goalY - (currY-1))  + L + B;

  // Same row as current position
  score3 = abs((goalX - (currX-1))) + abs(goalY -  currY  )  + R;
  score4 = abs((goalX - (currX+1))) + abs(goalY -  currY  )  + L;

  // One row above current position
  score5 = abs((goalX - (currX-1))) + abs(goalY - (currY+1)) + R + F;
  score6 = abs((goalX -  currX   )) + abs(goalY - (currY+1)) + F;
  score7 = abs((goalX - (currX+1))) + abs(goalY - (currY+1)) + L + F;


  // Initialize the outcome to score7 and choose Min of all possible
  // next positions
  myScore = score7;

  stepX = 1;
  stepY = 1;
  if (score6 < myScore) {myScore = score6; stepX =  0; stepY =  1;}
  if (score5 < myScore) {myScore = score5; stepX = -1; stepY =  1;}
  if (score4 < myScore) {myScore = score4; stepX =  1; stepY =  0;}
  if (score3 < myScore) {myScore = score3; stepX = -1; stepY =  0;}
  if (score2 < myScore) {myScore = score2; stepX =  1; stepY = -1;}
  if (score1 < myScore) {myScore = score1; stepX =  0; stepY = -1;}
  if (score0 < myScore) {myScore = score0; stepX = -1; stepY = -1;}

  // If obstacles are encountered in all directions, do not move.
  if (myScore > obstaclePenalty) {
    stepX = 0; stepY = 0;
  }

    // Store the directional vector times the number of steps to take.
  nextMoves[0] = stepX * numSteps;
  nextMoves[1] = stepY * numSteps;

}
```

Source Code (Lighting)

```
// ----------------------------------------------------
// Team members:  Tarek Mohktar, Paul Yanik
//
// Description:
// This design accommodates the sketch
// and actuages LEDs.
// ----------------------------------------------------

// ----------------------------------------------------
// Pin Assignments and Variable Declarations
// ----------------------------------------------------
// Pin assignments
int voiceBit0 = 0;
int voiceBit1 = 0;
int currMood = 0;
int happy = 0;
int sad = 1;
int angry = 2;
int afraid = 3;

// ----------------------------------------------------
// IO Pin Setup
// ----------------------------------------------------
void setup()
{
  pinMode(14, INPUT);
  pinMode(15, INPUT);

  for (int y=0; y<6; y++) {
    pinMode(y, OUTPUT);
  }

 // Open a serial link for onscreen variable monitoring.
  Serial.begin(9600);
}

// ----------------------------------------------------
// Loop for Runtime Operation
// ----------------------------------------------------
void loop()
{
  voiceBit0 = digitalRead(14);
  voiceBit1 = digitalRead(15);
```

```
  if ((voiceBit1 == LOW ) && (voiceBit0 == LOW )) {currMood = happy;}
  if ((voiceBit1 == LOW ) && (voiceBit0 == HIGH)) {currMood = sad;}
  if ((voiceBit1 == HIGH) && (voiceBit0 == LOW )) {currMood = angry;}
  if ((voiceBit1 == HIGH) && (voiceBit0 == HIGH)) {currMood = afraid;}

Serial.println(currMood);

  if (currMood == happy) {
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
  }

  if (currMood == sad) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    digitalWrite(6, LOW);
    digitalWrite(7, LOW);
  }

  if (currMood == angry) {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
    digitalWrite(7, HIGH);
  }

} // end
```

Discussion:

The ET robot system is capable of reaching any point (x,y) within a single room. Future work would include the ability of the robot to move between rooms and also to have a more deterministic motion plan. Slip in the belt drives of the system made reaching a precise point prohibitive. A chain drive or threaded drive axel would remedy this shortcoming.

It was originally postulated that the IR sensors used for obstacle avoidance would provide a distance measurement to near obstacles. Instead, these devices provide only a digital

output that indicates the presence or absence of an obstacle. Had these devices provided analog output, greater intelligence (fuzzy logic, etc.) in the adaptive motion of the robot would have been possible.

The project posed more possibilities in its future implementations than it addressed. In addition to being a simple object retriever, the device could serve as a personal mobililty support apparatus or emergency communication system. Or – as machine intelligence increases, the device could transform into a true companion capable of anticipation and intelligent personal interaction (conversation, etc.).