1 Overview

We propose to implement a microcontroller for a robot designed to locate a transmitter broadcasting at a given signal strength. To find the source, the robot will sample the signal with an omni-directional senson and make calculations based on the intensity of the signal and the robot's location to determine the position of the transmitter. Possible extensions include using a more mathematically complex search algorithm, dealing with obstacles in the robot's path, and locating a signal of unknown intensity.

2 Algorithm

In order to simplify the control of the robot, its operation will be restricted to two degrees of motion in discrete steps. In order to triangulate the source of the signal, the robot will store a set of values which represent the boundaries of the region in which the robot has determined the source to be. As the robot travels through this space, it continually shrinks the boundaries according to the intensity of the signal.

- Microcontroller State

5 bits	current intensity (32 levels)
5 bits	previous intensity
4 x 5 bits	boundaries (xmin, xmax, ymin, ymax)
2 x 5 bits	current location (x,y)
2 bits	direction (north 00, east 01, south 10, west 11)
2 bits	start direction
1 bit	wrong direction

- Initialization

  The robot begins by assuming it is in the middle of a 32 x 32 world. The source must be located somewhere in that world.

  set xmin = 0;
  set xmax = 31;
  set ymin = 0;
  set ymax = 31;
  set lastIntensity = 0;
  set currIntensity = readSensor();
  set currPosition = (15,15);
  set direction = north;

- Iterative Search

```
main:
    updatePosition();
    set lastIntensity = currIntensity;
    set currIntensity = readSensor();
    if (currIntensity > lastIntensity) /* Robot moved toward source. */
      /* Operation: Set boundary at previous location. */
      case (direction)
           north : set ymin = y(currPosition) - 1;
           south : set ymax = y(currPosition) + 1;
           east  : set xmin = x(currPosition) - 1;
           west  : set xmax = x(currPosition) + 1;
    else if (currIntensity < lastIntensity) /* Robot moved away from source. */
      /* Operations:
           1. Reverse direction
           2. Set boundary at current location.
           3. Set wrong direction bit. */
      case (direction)
           north : set direction = south;
                   set ymax = y(currPosition);
                   set wrongDirection;
           south : set direction = north;
                   set ymin = y(currPosition);
                   set wrongDirection;
           east  : set direction = west;
                   set xmax = x(currPosition);
                   set wrongDirection;
           west  : set direction = east;
                   set xmin = x(currPosition);
                   set wrongDirection;
    else /* Intensities were equal. */
      /* Operation: Execute a "right turn." */
      case (direction)
           north : set direction = east;
           south : set direction = west;
           east  : set direction = south;
           west  : set direction = north;
```

```
updatePosition:
  if (wrongDirection)
    move(direction);
    set wrongDirection = 0;
  if (insideBounds(direction)) /* If we haven't run into a boundary */
    currentPosition = move(direction); /* Move forward in current direction */
  else
    startDirection = direction /* Save this so we can check to see if we're boxed in. */
    direction = rightTurn(direction);
    while (direction != startDirection and !insideBounds(direction))
    /* While we haven't turned in a full circle and we can't move forward */
          direction = rightTurn(direction);
    if direction = startDirection
          /* We can't move and so we're done! */
          Exit();
    else
          move(direction);
```

3 Simulation code

Tested simulation code for this algorithm can be found at http://www.owlnet.rice.edu/ jessica/elec422/robot.cc .

4 Functional Units

1. Register Files

   The registers implemented will correspond to the microcontroller state. (See section 2.)

2. Programmable Logic Array

   The PLA will implement algorithm control for the robot. The following signals will be inputs and outputs of the PLA (this is still under debate):

In or Out	Symbol	Functional Name	Bits
O	ALUop	ALU Op Ctrl	1
I	ALUret	ALU Ret Sel	2
I	WDR	Wrong Dir Reg Dat	1
O	WDT	Wrong Dir Reg Tog	1
I	DA	Data Acquire	1
I	DR	Data Ready	1
O	DHR	Dir Heading Req	2
I	DHD	Dir Heading Data	2

O	WX	Write X Pos Reg	1
O	WY	Write Y Pos Reg	1
O	WXMI	Write XMin Reg	1
O	WYMI	Write YMin Reg	1
O	WXMA	Write XMax Reg	1
O	WYMA	Write YMax Reg	1

Also included will be several multiplexer control signals, as yet to be determined. (See Issues section below.)

3. Arithmetic Logic Unit

   The ALU will implement the following functions:

   (a) Add
   (b) Subtract
   (c) Less Than
   (d) Greater Than
   (e) Equal To

   These functions will be implemented using a full adder/subtracter and using the subtract to perform comparisons.

5 I/O Pins

Pins	Symbol	Functional Name
1	R	reset bit
1	Q	query bit
1	QP	query position bit (0 - X, 1 - Y)
2	QB0-1	query boundaries (00 - N, 01 - E, 10 - S, 11 - W)
5	S0-4	sensor input bits
2	D0-1	direction output bits (00 - N, 01 - E, 10 - S, 11 - W)
5	P0-4	position output bits
5	B0-4	boundary output bits
1	WD	wrong direction bits
1	DA	data acquire bit
1	DR	data ready bit
5	ST0-5	state output bits

6 Data Path

See attached diagram.

7 Project Home Page

Project home page is located at http://www.owlnet.rice.edu/ jessica/elec422