The Robot Builder

Volume Eleven Number Eleven November 1999

Notices

· Introductory Mobile Robotics Class - 10:00AM - 12:00PM

· Business Meeting - 12:30 - 1:00

· General Meeting - 1:00 - 3:00

Distribution

If you would like to receive The Robot Builder via e-mail, contact the editor at:

apendragn@earthlink.net

Inside this Issue

A Circular Navigation System 1

Polly 5

"A Circular Navigation System"

Part 3 - "Localization"

by Jim Ubersetzig

Introduction

Have you ever wanted your robot to navigate accurately? Reliably moving to locations within a room is difficult for most robots. Usually they are not aware of their location. In this series of articles, I will show you how to solve the problem for indoor robots.

Part one was the theory document and you should refer to it when you have questions about the mathematics. (see Sept 99 issue) Part two provided instructions for building the sensor unit, and included the I/O driver software. (see Oct 99 issue) This month we will add the capability to report the robots location. Accuracy is plus or minus a few inches within a typical room.

Skills Required

You will need certain minimum skills to complete the device described in this article:

Soldering, electronic grade.
Cutting plastic sheet.
Safety skills to avoid injuries.
Drilling holes.

Adding the Protractor Scale

Photocopy figure 1 and glue it to a thin sheet of cardboard. Punch a hole in the center and cut a slot to the edge. Slip the 3 inch diameter protractor under the disk of the top servo. Attach with mounting tape. Find a suitable material and attach a pointer for reading the angles.

Note that this protractor has 256 divisions in a complete circle. These divisions are called "binary angle measurement" (BAM) and are used by the Stamp II computer for the trigonometry calculations. One BAM equals approx. 1.4 degrees of angle.

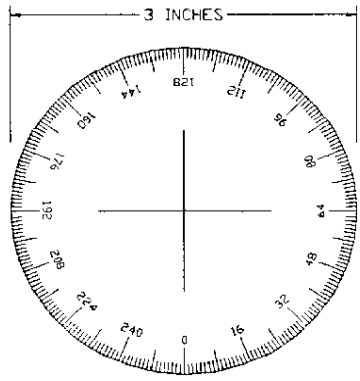

Figure 1
Binary Angle Measurement
Protractor Scale

Correction to Figure 6 of Part 2

In part two of this series of articles, a laser-power-switch was mentioned two paragraphs after figure 6. This switch was omitted from the wiring diagram in figure 6. Please install the laser-power-switch in series with the red wire of the laser module.

Analytic Geometry

Analytic Geometry is the bridge between the geometric argument stated in the 1993 theory document (reprinted in part 1 of this series of articles), and numerical calculations suitable for a computer.

The geometric figures of the theory document can be drawn to scale on graph

See *Circular*, Page 2

Circular from Page 1

paper. Doing so illustrates that for each point on the floor of the room, there are two numbers defining the location of that point. Since the robot is on the floor, our problem is to determine the two numbers defining it's location.

First we have to establish a coordinate system (see figure 2).

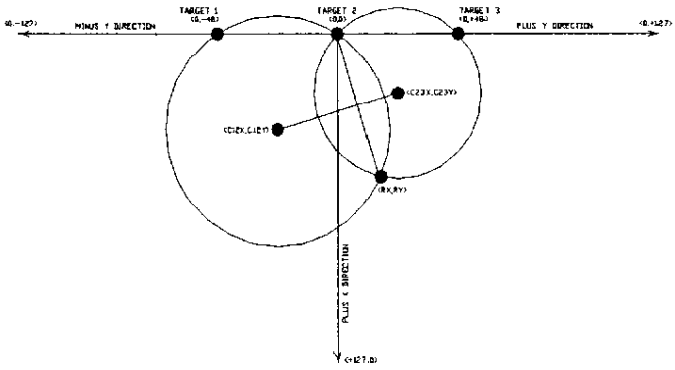

Figure 2 - Coordinate System

The solution space for the (rx,ry) robot position is: $0 < rx < 127$, $-127 < ry < +127$ where the unit is the inch. Note that rx is the distance from the wall, ry is the distance along the wall. This is approximately a 10 by 20 foot room. Target 1 is at (0,-48), target 2 is at (0,0), target 3 is at (0,+48).

Per the theory document, the robot is at the crossing of two circles. C12 at position (C12X,C12Y) is the center of the circle which passes through targets 1 and 2 and the robot position. C23 at position (C23X,C23Y) is the center of the circle which passes through targets 2 and 3 and the robot position.

The laser scanner head measures angles A12 and A23 in units of 1.406 degree. A12 is the angle between targets 1 and 2 as seen from the robot position. Similarly A23 is the angle between targets 2 and 3.

The Method - General

The solution proceeds as follows: First a table lookup finds the centers of the circles from A12, and A23. Next the equation for a line is found in slope, X intercept form.

This is for the line passing through C12 and C23. The argument is made that this line is the perpendicular bisector of the line segment from (0,0) to (rx,ry).

We proceed to find the intersection of these lines, then from the known endpoint (0,0) and the bisector, we find the other end of the line segment at (rx,ry), which is the solution.

The Method - Details

Subroutine get_cx performs the table lookup, this subroutine is used twice, to compute C12X and C23X. The Y coordinates C12Y and C23Y are precalculated and stored as constants.

We want the equation of a line passing through circle centers C12 and C23. Because the line always crosses the x axis, we want the line's equation in slope, x intercept form. The equation is

$$x = my + Xint,$$

where the slope is:

$$m = \frac{C23X - C12X}{C23Y - C12Y} = \frac{N \text{ (numerator)}}{D \text{ (denominator)}} \quad \text{(equation 1)}$$

and

$$Xint = (C23X + C12X)/2 \quad \text{(the X intercept)}$$

Note that D can be precomputed and = Ta12, which is the spacing between targets.

The other line we are interested in is the line from (0,0) to the sensor assembly at (rx,ry). Since this line is perpendicular to the other line, their slopes are related by $m = -1 / m$. Because the line passes through (0,0), the X intercept = 0.

And the equation is

$$x = \frac{-1}{m} y$$

We want the point where the two lines cross, so we combine the equations:

See *Navigation*, Page 3

Navigation from Page 2

$$\frac{-1}{m} \quad y = my + Xint$$

or

$$-2y = \frac{N * Xint2 * Ta12}{(TaSq + N^2)} = \frac{top}{bottom} \qquad \text{(equation 2)}$$

where N = C23X - C12X, Xint2 = 2 * Xint = C23X + C12X, and TaSq = Ta12 * Ta12.

-2y is the sensor assembly position ry coordinate. the corresponding rx coordinate is:

$$rx = 2x = -2y \frac{Ta12}{N}$$

There remain a few tricks required to get the correct answer from the Basic Stamp 2 computer. First, division only works if both numbers are positive. For calculating -2y, the bottom is always positive. We save the sign of the top and then make it positive, then divide by bottom to get 2y. We restore the sign of 2y later. Second, the Basic Stamp 2 uses 16 bit integer arithmetic. Because the numbers top and bottom can be quite large - they can sometimes be larger than the maximum 16 bit number. Also the result dividing top by bottom has precision only if top is substantialy larger than bottom.

Avoiding these problems of integer arithmetic requires careful management of the top and bottom numbers. This is done by pre-divided top and bottom to make them the correct size for an integer division.

Loading the Software

On your PC type stamp2 <enter> and the stamp development system will come up. If you don't have this file (stamp2.exe approx 15K bytes), it's a free download from the parallax web site. It also comes with the developer's kit, along with the cable.

If you don't have the cable, instructions for building one are on the web site. On your PC you should see a screen for editing software. Type in the main software listing (see above). Be sure to include the software from part 2 of this series of articles - the location for this is clearly marked.
Press ALT-S to save the software in a file. Type in a suitable file name, then <enter>.

Demonstration

Tape three retroreflective targets on a wall. Space the retroreflectors every 4 feet. Adjust the target height to match the laser beam. The laser beam must be parallel with the floor - adjust if necessary. Press ALT-R on the PC. After the laser sweeps in a complete circle, you should see a report on the PC display screen.

Hit_index should be three (the number of targets found). The number reported for RX is the distance from robot to the wall. RY is the distance along the wall measured from the middle target. Sign of zero means that the robot is to the right of the center target. Else the robot is to the left.

Error Messages

Sometimes instead of reporting the robot's position, an error message appears. There are two kinds of errors reported:

Cause	Error Message	RX value
Object blocks laser beam.	Lost - can't find 3 targets.	-1
Targets on two walls.	Lost - targets on more than one wall	-2

If you choose to use this sensor on a robot, you can write additional software which directly uses the values of RX, RY, Sign. Check RX first - if RX.bit7 = 0, then the position reported is valid. Else check for RX = 0xFF (-1) or RX = 0xFE (-2) to determine the error.

The Future ?

Next month we will add the capability to navigate a robot from it's present location to any desired location. Accuracy is plus or minus a few inches within a typical room.

Jim Ubersetzig
jim.ubersetzig@lmco.com
(661) 572-7184

(Code is continued on page 4)

```
' Software to report the robots location.
' Accuracy is plus or minus a few inches within a typical
room.
' Software runs on a Stamp 2.

' By Jim Ubersetzig
' May 99

        gosub locate

spin:   GOTO spin

'--------------------------------
' Insert software from part 2   |
' of this construction article. |
' ( must end with return )      |
'--------------------------------

locate:
   gosub scan ' perform a complete laser scan of the room
              ' and measure the angles to the wall targets.

' code to decide if data from sensor head is sufficient
if hit_index = 3 then data_correction
   debug "lost - can't find 3 retroreflector targets",cr
   rx  = -1
   goto locate2

data_correction:
for hit_index = 0 to 2     ' correct the measured angles
   hit(hit_index) = hit(hit_index) * 64 / 50
next

A12     var     byte
A23     var     byte

' code to decide which is target 1, target 2, and target 3
' and to calculate A12, A23 in units of 1.406 degrees.
        if (hit(1) - hit(0)) < 128 then zzz1
        A12 = hit(2) - hit(1)
        A23 = hit(0) - hit(2) // 256
        hit = hit(2)       ' save angle to target 2
        goto locate1
zzz1:
        if (hit(2) - hit(1)) < 128 then zzz2
        A12 = hit(0) - hit(2) // 256
        A23 = hit(1) - hit(0)
        hit = hit(0)       ' save angle to target 2
        goto locate1
zzz2:
        if (hit(0) - hit(2) // 256) > 128 then zzz3
debug "lost - targets on more than one wall.",cr
        rx = -2
        goto locate2
zzz3:
        A12 = hit(1) - hit(0)
        A23 = hit(2) - hit(1)
        hit = hit(1)       ' save angle to target 2

locate1:
debug ? A12, ? A23

' these distances are in inches.
Ta12    con    48    ' distance between target 1 and target 2.
Ta23    con    Ta12  ' distance between target 2 and target 3.
C12Y    con    -Ta12 / 2 ' y value for center of circle C12.
C23Y    con     Ta12 / 2 ' y value for center of circle C23.
TaSQ    con     Ta12 * Ta12 ' square of Ta12.

RX    var   byte   ' robot position - distance from wall.
RY    var   byte   ' robot position - distance along wall.
Sign var    bit    ' 0 -> + = to right of target 2.

SigBits    con   9          ' bits of precision for result of
                            ' integer division.

' solve for robot location (rx,ry)

' reuse these words of ram
C12X    var    angle     ' x value of circle center C12.
C23X    var    reflection    ' x value of circle center C23.
   RY = A12: A12 = A23: gosub get_CX: C23X = C12X
   A12 = RY: gosub get_CX     ' calculate centers for both
                              ' circles.

' reuse these words of ram
N    var    C23X     ' numerator from equation 1.
Xint2 var C12X     ' 2 times the X intercept.
   N = C23X - C12X
   Xint2 = N + C12X + C12X
   Sign = N.bit15 ' save sign (+/-) for N
   N = abs( N )   ' must be + for division to work.

   if N <> 0 then continue
      ry = 0: rx = Xint2: goto done' special case if robot
                                   ' is on x axis.
continue:

' reuse this nibble of ram
log    var    hit_index
   log = NCD( N * N + TaSQ ) + SigBits - 16  ' to prevent
                                             ' overflow.

' reuse this word of ram
top  var    Xint2    ' word
   top = Ta12 * Xint2 / DCD(log) * N    ' top from equation 2.
   ry = top / ( N * N + TaSQ / DCD(log) ) ' top / bottom.
   rx = ry * Ta12 / N
   Sign = ~Sign             ' robot is at rx, ry, sign of ry

done:
' solution for robot location is (RX, RY)
debug sdec ? RX, sdec ? RY, ? Sign

locate2: return


' This code gives the correct value for C12X (in inches)
' for 1 <= A12 <= 127 (in units of 1.406 degrees).
' Assumptions are Ta12 = 48 inches.
get_cx:
   if A12 < 65 then get_cx1
   read  128-A12 * 2, C12X.highbyte
   read  128-A12 * 2 + 1, C12X.lowbyte
   C12X = -C12X
   goto get_cx2:
get_cx1:
   read  A12 * 2, C12X.highbyte
   read  A12 * 2 + 1, C12X.lowbyte
get_cx2:
   return

 data    0,0, 3,210, 1,233, 1,69, 0,244, 0,195, 0,162 '0-6
 data    0,138, 0,121, 0,107, 0,96, 0,87              '7-11
 data    0,79, 0,73, 0,67, 0,62, 0,58                 '12-16
 data    0,54, 0,51, 0,48, 0,45, 0,42, 0,40           '17-22
 data    0,38, 0,36, 0,34, 0,32, 0,31                 '23-27
 data    0,29, 0,28, 0,26, 0,25, 0,24                 '28-32
 data    0,23, 0,22, 0,21, 0,20, 0,19                 '33-37
 data    0,18, 0,17, 0,16, 0,15, 0,14                 '38-42
 data    0,14, 0,13, 0,12, 0,11, 0,11, 0,10, 0,9, 0,9 '43-50
 data    0,8, 0,7, 0,7, 0,6, 0,5, 0,5, 0,4, 0,4       '51-58
 data    0,3, 0,2, 0,2, 0,1, 0,1, 0,0                 '59-64

' End of the software
```

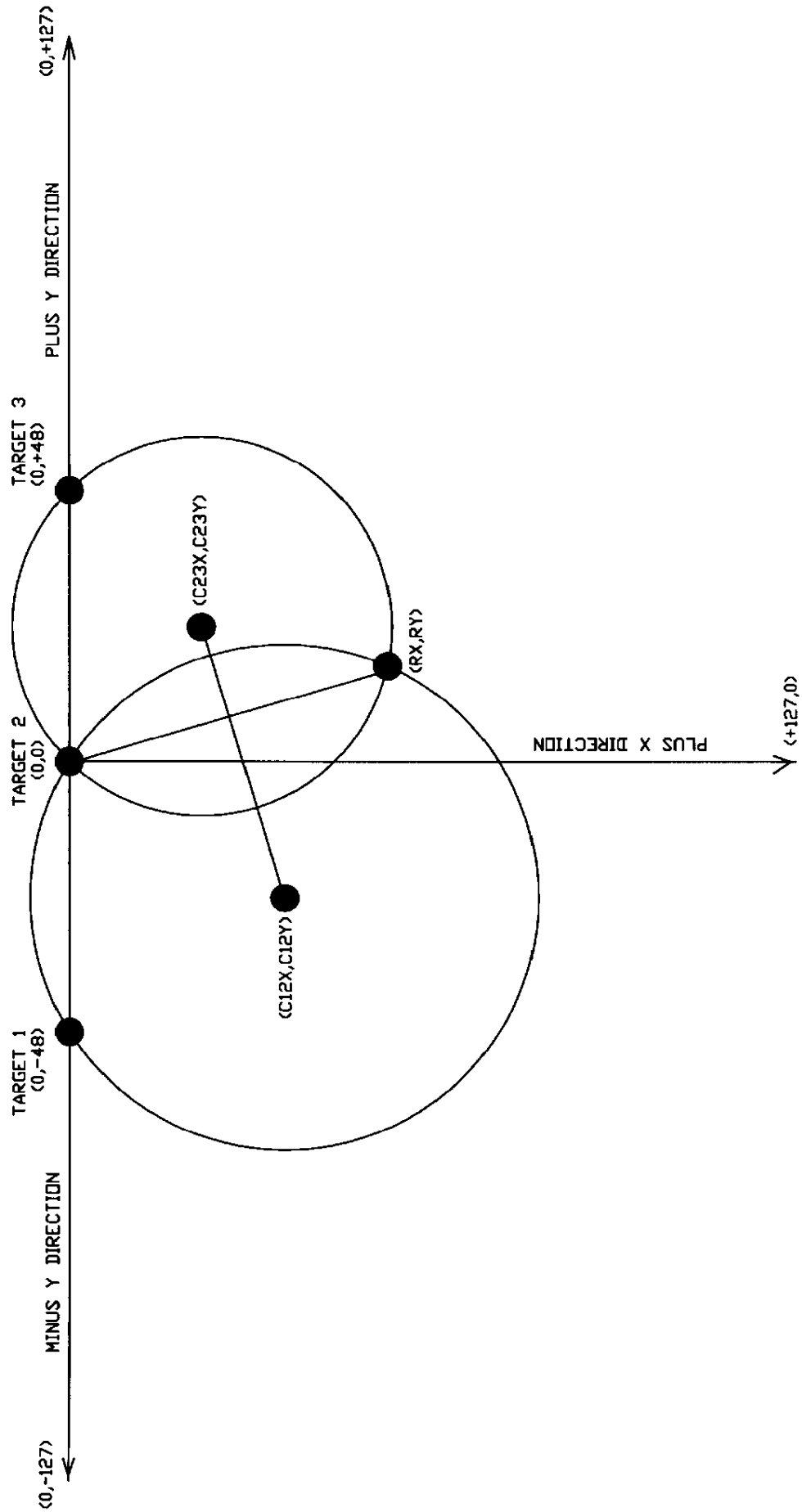

Figure 2 - Coordinate System