The Robot Builder

Volume Eleven Number Four April 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

DUAL SENSORS for a STAMP (Part II)…….…. 1

Introduction to State Machines……..……. 5

Dual Sensors for a Stamp

Part II

by Jim Ubersetzig

Introduction

Have you ever wanted to add sensors to your robot - but weren't sure how to do it ?

In this two part article I will show you how to build a pair of inexpensive Sonar and Infra Red (IR) sensors which share a Basic Stamp 1 computer. By operating both sensors off the same stamp computer - you save cost.

Last month we built the Sonar sensor and demonstrated its ability to measure the distance to an object. This month we will build an infra red (IR) sensor which reports the probability of objects to the right, to the left and directly ahead of the robot.

The IR sensor reports confidence numbers, which are probabilities in the range 1 through 10. If the right confidence number is 10, the path is clear to the right. A high left number indicate confidence that the robot could turn left without striking an obstacle. When both left and right report 10, the robot can safely roll forward.

Parts and Cost

You have the choice of building the IR sensor assembly from parts you purchase or building it from a kit. The design of the IR sensor assembly is from "Standard Technologies of the Seattle Robotics Society" and can be found at:

www.seattlerobotics.org/guide/infrared.html

An almost identical sensor assembly is available as a kit from Lynxmotion. The kit has the advantage of a printed circuit board - which simplifies assembly. In addition to the Basic Stamp 1, and 9 volt battery you purchased for part one of this article, you will need either:

IRPD-01 kit (approximately $35) from
Lynxmotion
www.lynxmotion.com
(309)382-1816

or else purchase this list of parts and build it yourself. Costs given are approximate:

Qty	Part No.	Description	Cost
1		Perf Board	1.00
1		10K trim pot	1.00
1	G1U52X	Sharp IR module	5.80
1	74HC04	Hex Inverter	1.75
2		270 ohm resistor	0.14
2		IR LED, clear	1.20
1		3.3K resistor	0.07
1		10K resistor	0.07
1		100K resistor	0.07
1		0.001**uF** capacitor	0.25
1		0.1**uF** capacitor	0.55
		total	11.90

To load the software in this article into the Basic Stamp, you need a PC. The stamp also uses the PC to display test results. If you use the remaining Basic Stamp pins to build a robot base to mount the sensor on, then the robot could operate without the PC.

Skills Required

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

(see Sensor on page 2)

President's Message
April 1999

Happy Spring!

This has been an interesting month in the local robotics community. The legal differences between Profile Records and Marc Thorpe has finally been resolved. Profile has purchased the rights to the RW name and Marc will continue to receive a percentage of the profits from future RW events. The 1999 Robot Wars event is scheduled for August 20-22 at the Fort Mason site. Information can be found at the RW Website at http://www.robotwars.com

On a similar note, a NEW Robotics Combat event has been announced for this August. BATTLEBOTS is a 2 day event scheduled for August 14-15, 1999 at the Pyramid at Cal State Long Beach. This event is being run by Trey Roski, the driver of La Machine. Information can be found at the Battlebots website at http://www.battlebots.com

Bot Bash4 1999 is scheduled for July 11 at the Hexacon Science Fiction convention in Scottsdale, Arizona. There will be 5Kg, 12.5Kg, and 23Kg weight classes and three categories of teleoperated competition, the Obstacle course, Sumo fight, and Robot battle. There will also be a separate autonomous competition as well. Information on Bat Bash4 1999 can be found at their website at http://www.primenet.com/~johnkit/bash.html

I hope to be attending all of these events and look forward to seeing some of you there. Registration and entry information can be found at the individual sites. These competitions are only 3-4 months away, so get started on your projects YESTERDAY! The competition will be fierce. This looks to be an exciting summer for robotic combat indeed.

That's about it for this month, now let's get out there and build something!

Randy Eubanks
President
Robotics Society of Southern California

Sensor from Page 1

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

Construction - Kit

If you build the IR board from the kit, the manual is excellent. You might be able to set the trim pot without measuring the frequency - the correct setting for the author's kit was 1:45pm (noon being half way).

Construction - Do It Yourself

Begin by cutting a piece of perfboard at least 1-1/2" by 2-1/4". Mount the Sharp receiver module at the center of a long side of the board. Solder a short wire from the ground pin to the metal case.

Now mount the IR LEDs on both sides of the receiver module. Leave the leads long and bend them at 90 degrees so the IR LEDs and the receiver module point in the same direction. Next angle the LEDs out at about 30 degrees to the left and right. (see figure 3)

The rest of the parts can be mounted anywhere on the perfboard. Wire the transmitter section as shown below:

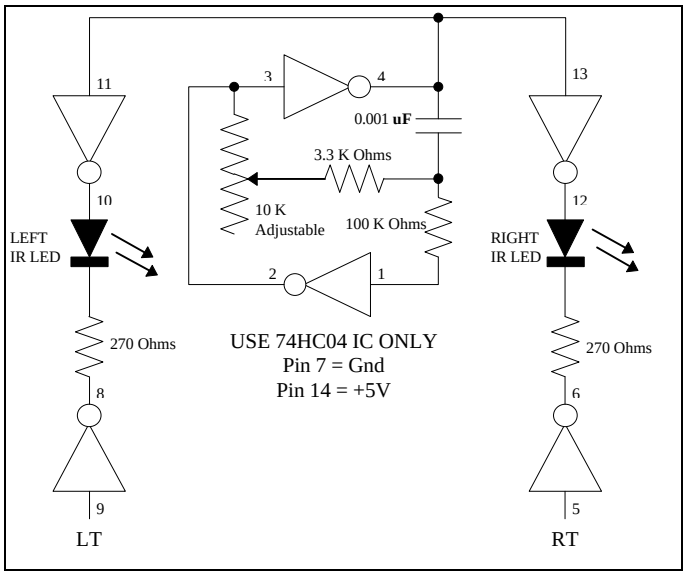

Figure 1 - Schematic Left and Right IR Transmitter

Now wire the receiver as shown. Mount the 0.1 uF capacitor close to the receiver module. Bend the capacitor leads and solder directly to the ground and +5V pins of the receiver module.

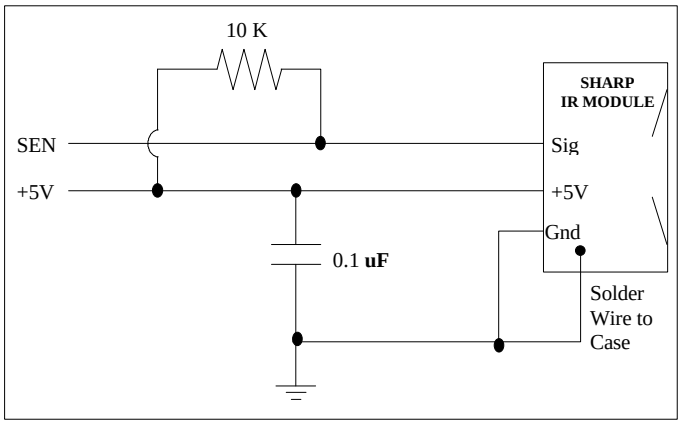

Figure 2 - Schematic of IR Receiver

(see Sensor on page 3)

Sensor from Page 2

Theory of Operation

The IR sensor operates by bouncing infra red (non visible light) off objects. Two IR transmitters produce brief bursts of light, while a IR receiver looks for reflections from nearby objects.

The electronics works like this: Two sections of the 74HC04 are used to form an oscillator. The 10K ohm trimpot is for adjusting the operating frequency to 40KHz. The alternating voltage from the oscillator is applied to both IR LED's. A voltage is applied to the other side of each LED to select which LED emits light.

The IR receiver module was designed for IR remote controls, so the oscillator must be tuned to match the required 40KHz light flash rate. IR is radiant heat, and most objects don't change temperature 40,000 times per second, so hot objects do not affect the IR receiver. However sunlight contains LOTS of IR, and will stop the IR sensor from working.

The optical components are mounted with overlapping fields of view to distinguish between objects to the right, to the left and straight ahead.

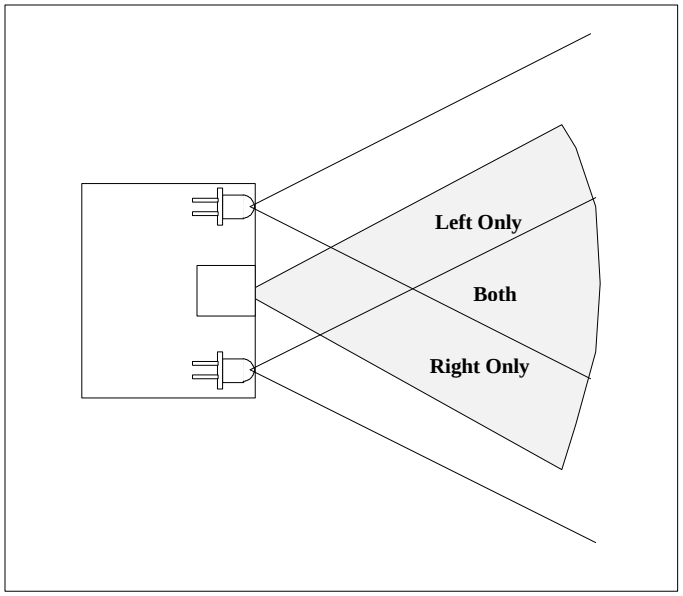

Figure 3 - IR Board Sense Zones

In this diagram, you are looking down at the robot. The software leaves the left IR LED off, and flashes the right LED ten times, counting the number of of received flashes. Then the process is repeated with the right LED off, flashing the left LED ten times. Confidence numbers are reported for right and for left.

The secret of getting a reliable reading for directly ahead of the robot is careful alignment of the IR LEDs. Then when both right and left confidence numbers report 10, the robot is clear to roll forward.

There are a couple of things to keep in mind when using the confidence numbers:

1. Sunlight from a window will prevent operation of the sensor.
2. When both numbers are large (9 or 10), the robot can roll forward.
3. With right number large and left number small, the robot should turn right.
4. Left larger than right means turn left.
5. If both numbers are small (1 to 2). the robot should back up.

The Cable

You need to build a cable to connect the Basic Stamp 1 to the IR board. Build it to match the wiring diagram:

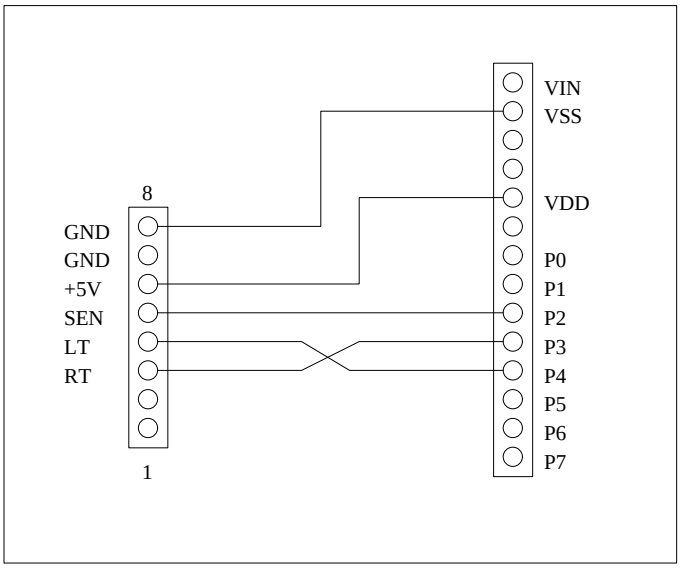

Figure 4 - IR Sensor to Stamp 1 Cable

(see Sensor on page 4)

Sensor from Page 2

Software

Here is the Basic Stamp code used in this project:

```
'
' Test software for the Infra Red sensor
' derived from the IRPD manual.
' 31 May 98


' Runs on the Basic Stamp BS1-IC
'

' BS1-IC    IRPD   function
' Pin       Pin
' P2        5      IR sensor input
' P3        4      Look Right
' P4        3      Look Left
' Vdd       6      +5VDC
' Vss       8      ground

' symbols
symbol temp = b2
symbol x = w2

dirs = %00011001     ' P0,P3,P4 = are outputs, others are inputs

start:    temp = 0
  for x=1 to 10             ' do this ten times
    pin3 = 1                ' shine IR light to the right
    pin3 = 0                ' turn it off
    temp = temp + pin2      ' count the misses
  next x

  debug "right ",#temp,cr   ' display 0-3 for obstacle,
                            ' 7-10 for nothing.
  for x=1 to 750 :next      ' delay for reading display.

  temp = 0
  for x=1 to 10             ' do this ten times
      pin4 = 1              ' shine IR light to the left
      pin4 = 0              ' turn it off
      temp = temp + pin2    ' count the misses
  next x

  debug "left ",#temp,cr    ' display 0-3 for obstacle,
                            ' 7-10 for nothing.
  for x=1 to 750 :next      ' delay for reading display.

  goto start

' End of the software
```

Get It Working ?

On your PC type stamp <enter> and the stamp development system will come up. If you don't have this file (stamp.exe approx 15K bytes), it's a free download from the parallax web site at **www.parallaxinc.com**. 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 blue screen. Type in the software, then ALT-S to save the software in a file. Type in a suitable file name, then <enter>. Now ALT-Q to exit. The software is saved on your PC.

To operate the IR sensor:
On the PC type stamp <enter>
You'll see the blue screen again. ALT-L will let you select the software you have previously saved.

Once you see the software on the PC display, hook up the cable from your PC's com port to the stamp. NOTE that the cable can go two different ways on the stamp. Line up the marks !

Now ALT-R will start the software running. You should see a list of numbers on the computer screen. These are the confidence numbers reported by the IR sensor.

Place an object where the sensor should see it and the numbers should change. Experiment with different objects and different distances.

Troubleshooting

If things don't work as expected, there are adjustments:

First, check the software you typed in. Did you copy it exactly ?

If all the numbers are zero - the receiver is viewing the transmitting IR LEDS. Bend the leads of the LEDS so the lens of the LED is behind the front face of the receiver module. Or place cardboard between the transmitters and the receiver.

If the numbers reported vary widely (or are always 10), the oscillator requires adjustment. This can be measured with a frequency counter, some digital voltmeters have this built in. Alternately you can measure the period with a scope. The period for 40KHz is about 25 microseconds.

If the sensor is too sensitive (detects objects too far away), place layers of clear adhesive tape over the opening in the sharp IR receiver module. The author found that six layers of scotch tape were best.

The Future ?

Many improvements are possible, and the author encourages experimentation. Try things. Some of your ideas will work.

Jim Ubersetzig

juberset@lmco.com

Introduction to State Machines

by Arthur Ed LeBouthillier

There are many different programming techniques which one uses only infrequently. One technique, however, can be used repeatedly and regularly: the State Machine.

State machines represent one of the most fundamental principles of computing and programming. They represent the fact that a system can exist in certain states and can transition from one state to another. The state machine is so universal of a concept that virtually all computing can be modeled by state machines. Everything from computers to the very programs that run on them are state machines.

State machines are very important in all aspects of the design of programs. They help gather your thoughts on a topic by providing a notation to represent many complicated interacting states. They also permit you to organize those thoughts into a coherent plan of action. Finally, because they are so easy to implement in code, knowing how to represent your problem as a state machine provides an immediate means to implementing them.

Introduction

A State Machine can be viewed as being composed of nodes representing possible states and conditional transitions between state nodes. Therefore, to represent a state machine, we use state nodes which represent each particular state and conditional transitions between them:

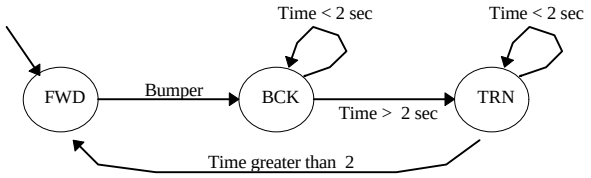

Figure 1 - A sample State Machine

In figure 1, a state machine is shown for a simple robot's motion. The state machine consists of three states: FWD, for forward motion; BCK, for backward motion; and TRN for turning. There is one entry transition pointing to the FWD state and four conditional transitions between each of the three possible states. A robot controlled by this state machine would enter into the FWD (or forward) state. If a bumper switch was hit, then the robot would transition to the BCK state and stay there as long as the time variable was less than 2 seconds. After 2 seconds at the BCK state, the state machine would transition to the TRN state, stay there for 2 seconds and then transition to the FWD state again, starting the whole cycle over. This state machine, although very simple, is representative of the state machine idea.

Using State Machines in the Conceptual Stage

State machine can help simplify the design of programs because they provide an easy notation for what is going on inside your robot. Gather together the different actions that might occur and call them the states. Gather a list of situations which would cause the system to transition from one state to another. In some cases, you will find that certain states don't belong together and that you might have different state machines instead of the one that you originally thought you had.

It is possible to use multiple state machines that enable each other. Using the state machine in figure 1, we might have other state machines which are enabled in each of the three states above. For example, while going forward, we might beep and go forward. While backing up, we might beep, flash a light and then back up. While turning we might flash the lights. The way this would be implemented is by having a total of four state machines, of which the operation of three are controlled by the state machine above. This is illustrated in figure 2. The double circles with the "End" name are the final states of each state machine. When each state machine enters this state, it does not leave.

Implementing State Machines

Implementing state machines once you have the state machine diagrams is easy. For each state machine that is active, you need a variable to represent its state. In the above example, we would only need 2 variables to keep track of the state

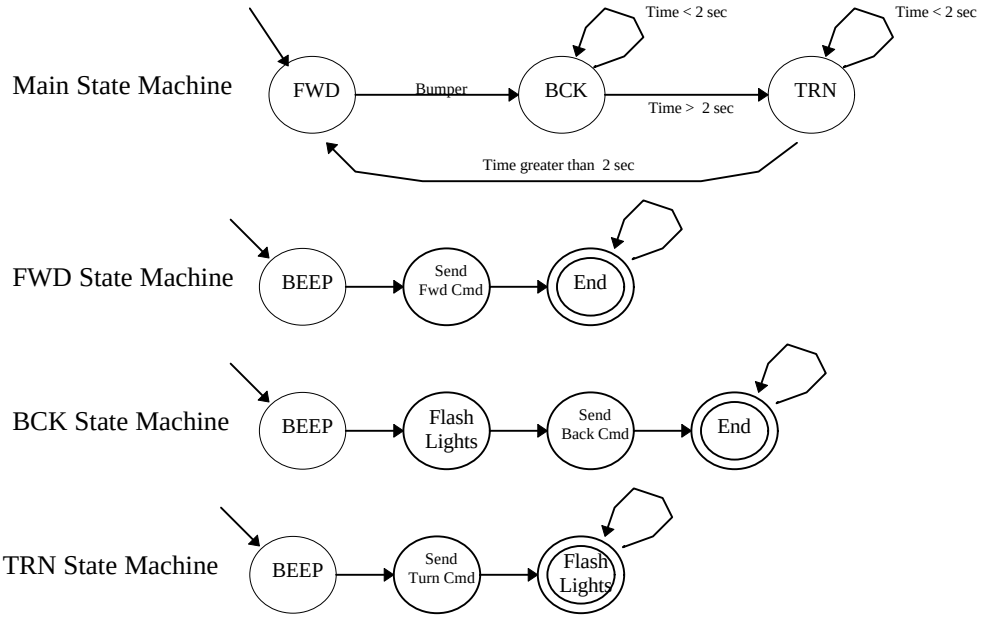

Figure 2 - A more advanced state machine

machines since only the main and one substate is active at a time. State machines are usually implemented as "**switch**" or "**select**" statements (depending on the language you are using) inside of a continuous **while** loop. Assign a number to each state in each state machine. You can restart at 0 for each different state machine. A simple program to implement the state machine above is illustrated in Figure 3. You can put "action" code, or the code that is necessary to perform the state's action at the conditional statement used to set up the transition.

Conclusion

The state machine concept is very powerful and helps you both in the design and implementation stages of your program. It helps you organize many complicated events and conditions by providing a notation to detail your program's operation. It also has a very simple implementation structure which can be duplicated in every implementation.

```
Const FWD = 0
Const BCK = 1
Const TRN = 2

Dim MainState as integer
Dim SubState as integer
Dim theTime as integer

MainState = FWD
SubState = 0

Beep
SendMotors( FORWARD )
while TRUE
            select case MainState
                        case FWD
                                    select case SubState
                                                case 0
                                                            Beep
                                                            SubState = 1
                                                case 1
                                                            SendMotors(FORWARD)
                                                            SubState = 2
                                                case 2
                                                            SubState = 2

                                    end select
                                    if( bumper = 1)
                                                theTime = Now()
                                                SubState = 0
                                                MainState = BCK
                                    end if
                        case BCK
                                    select case SubState
                                                case 0
                                                            Beep
                                                            SubState = 1
                                                case 1
                                                            FlashLights()
                                                            SubState = 2
                                                case 2
                                                            SendMotors(BACKWARD)
                                                            SubState = 3
                                                case 3
                                                            SubState = 3
                                    end select
                                    if( DiffTime(theTime,Now()) > 2 )
                                                theTime = Now()
                                                SubState = 0
                                                MainState = TRN
                                    end if
                        case TRN
                                    select case SubState
                                                case 0
                                                            Beep
                                                            SubState = 1
                                                case 1
                                                            SendMotors(RIGHT)
                                                            SubState = 2
                                                case 2
                                                            FlashLights()
                                                            SubState = 2
                                    end select
                                    if( DiffTime(theTime,Now()) > 2)
                                                SubState = 0
                                                MainState = FWD
                                    end if
            end select
end while
```

Figure 3 - Code for state machine in figure 2.

Robotics Society of Southern California

President	Randy Eubanks
Vice President	Henry Arnold
Secretary	Arthur Ed LeBouthillier
Treasurer	Tom Thornton
Past President	Jess Jackson
Member-at-Large	Tom Carrol
Member-at-Large	Pete Cresswell
Member-at-Large	Jerry Burton
Faire Coordinator	Joe McCord
Newsletter Editor	Arthur Ed LeBouthillier

The Robot Builder (TRB) is published monthly by the Robotics Society of Southern California. Membership in the Society is $20.00 per annum and includes a subscription to this newsletter.

Membership applications should be directed to:

Robotics Society of Southern California
Post Office Box 26044
Santa Ana, CA 92799-6044

Manuscripts, drawings and other materials submitted for publication that are to be returned must be accompanied by a stamped, self-addressed envelope or container. However, RSSC is not responsible for unsolicited material.

We accept a wide variety of electronic formats but if you are not sure, submit material in ascii or on paper. Electronic copy should be sent to:

apendragn@earthlink.net

Arthur Ed LeBouthillier - editor

The Robotics Society of Southern California was founded in 1989 as a non-profit experimental robotics group. The goal was to establish a cooperative association among related industries, educational institutions, professionals and particularly robot enthusiasts. Membership in the society is open to all with an interest in this exciting field.

The primary goal of the society is to promote public awareness of the field of experimental robotics and encourage the development of personal and home based robots.

We meet the 2nd Saturday of each month at California State University at Fullerton in the electrical engineering building room EE321, from 12:30 until 3:00.

The RSSC publishes this monthly newsletter, The Robot Builder, that discusses various Society activities, robot construction projects, and other information of interest to its members.

Membership/Renewal Application

Name _____

Address _____

City _____

Home Phone ( ) - Work Phone ( ) -

Annual Membership Dues: ($20) Check #
(includes subscription to The Robot Builder)

Return to: RSSC
POB 26044
Santa Ana CA 92799-6044

How did you hear about RSSC? _____

RSSC
POB 26044
Santa Ana CA 92799-6044

Please check your address label to be sure your subscription will not expire!