Wireless Networks for Multi-Robot Communications Team

Clay Oehlke Matt Crotts Kenny McNutt Jeremy Vernon

Introduction

In this critical design review, we give an overview of the progress of our project, the wireless multi-robot communications. In this, we cover what we have already done, and show what we have left to do. We will also show some experimental results from the Boe-Bots themselves.

Navigation

In order to have the robots navigate around the arena, whiskers will be used. The whiskers are bumper switches. They are wired, with a resistor, to a high voltage source. When the whisker is untouched, the circuit is broken, and the BASIC stamp records a voltage of 5 Volts, or a logic high. When the whisker touches something, it completes the circuit, sending the voltage down to 0 Volts, or a logic low. Thus, obstacles can be detected by checking the whiskers input values.

With this information, we can navigate around obstacles. This can be done by the following algorithm:

1. Check the status of both the whiskers.
2. If either of the whiskers register low, back up some distance
3. If the left whisker is low, turn right. If it is the right, turn left.
4. If both whiskers are on, turn around

This algorithm works well for most cases. The only problem we may have is that if a Boe-Bot is trapped in a corner, turning in either direction will cause the robot to hit that side's whisker. This will cause it to turn in the other direction. This will cause the other whisker to be hit, resulting in an endless loop.

A solution to that might be to institute a right hand rule for collisions. Instead of backing up and turning around, collisions with either feeler would cause the Boe-Bot to back up, and turn to the right. This would be less efficient, as it would turn right at all barriers, and this would cause the robot to stray off of any search pattern.

Light Detection

All Boe-Bots, as part of the Boe-Bot kit, come standard with photoresistors. These devices have a resistance to electricity depending on the level of light they are exposed to. For high light levels, the photoresistors have a small level of resistance, for lower light levels, more. A basic RC circuit can measure the resistance of an element in a circuit. Using the resistors and capacitors found in the kit, we can build one.

The BASIC chip comes with a built in function that allows us to time a voltage discharge of an RC circuit. This allows us to come up with a value directly proportional to the light level the photoresistor is currently looking at. Using these relative values, we can judge the difference between monitored light values on either side of the Boe-Bot. From there we can turn the Boe-Bot so it faces the light. Add in the ability to move forward, and you are able to move toward a light source.

This is covered in the robotics manual that comes with the kit. In order to read the resistance value, the circuit is set to high, allowed to reach the 5 Volt level, then set back to zero. The time with which this takes is then reported. To use this, photoresistors are set on either side of the Boe-Bot, similar to the feelers. Values for each side are taken, then compared. The Boe-Bot then drives either left or right, depending on the higher side value. There is a deadband parameter though, a range of values for which the light strengths can be close to each other. This allows for the Boe-Bot to go forward if the photoresistor values are very close. The algorithm looks like this:

1. Determine values of left and right photoresistors.
2. If the left photoresistor reports a value that is some deadband value greater than the right, go right. Similarly, a larger right value entails going left..
3. If the two values are within the deadband value, then go forward.

Detecting Boe-Bot

Once the Boe-Bot has found the light, it must provide some means for other Boe-Bots to find it. We have decided to use the included infrared detectors for this purpose.

The detectors were originally designed to aid in object detection. The kit includes an infrared LED, and two infrared detectors. These would be used, like the feelers, to determine whether an object was in the path of a robot. A pulse would be sent out, and would bounce off any objects currently in front of the robot. This would be registered,

and the robot would drive around the obstacle. For our purposes, instead of a pulse, a long stream of infrared light will be emitted.

When one of the robots finds a suitable light source, it will emit the pulse. The other robots, using techniques similar to finding the original light source, will seek it out. As the IR detectors are less sensitive than the photoresistors, having only binary values of detecting infrared / not detecting infrared, the algorithm will have to be modified:

1. Robot finding light source will pulse using the infrared LED.
2. All other robots will check the values of their IR detectors
3. If both detectors return logic low or active high, go forward.
4. If either the left or right detector returns active high, and its counterpart returns active low, turn in the direction of the detector that is active high.

This might be a bit crude in the granularity of its searching ability, but as there will only be one light shining at a time, this should not be a problem. The goal is not a specific point, but a general area in which the Boe-Bot is.

Search Algorithm

In order to find a light source, all of the previous systems must be brought together. Also, a search algorithm must be developed to find the light source. As there will probably be multiple light sources, we must be able to:

1. Find a light source

2. Communicate this value and compare it to the other sources.
3. If the Boe-Bot has the highest value, start the IR LED
4. If it did not have the highest light value, search for the IR light

To find a light source, the light finding algorithm can be used, but not it alone. In order to find the source, the bots must do some random walking. This might be in the form of several random steps in a direction. The algorithm would look like this:

1. Pick a random direction
2. Drive a distance in that direction
3. Pick a random direction that was not the first
4. Go in that direction.
5. Rotate until a high enough light source was found
6. Use the light source finding algorithm.
7. When the values on the photoresistor reach a high enough threshold, report that you have found a light.
8. Wait a set amount of time until any other Boe-Bots report their light levels.
9. Use the IR detection to travel towards the highest light value.

Networking

In order to have the robots communicate with each other, a token ring network will be used to transmit the data. In a token ring network, only one node of the network can speak at a time, the node that contains the token. The token is passed from node to node on a path, and at the end of the path, it loops back to the start, forming the ring.

Several problems are inherent with token networks. In order to make a token ring network, a token must be created by one of the nodes. In our case, each robot will be programmed to listen for any one of the other robots to pass a token. If this does not happen, then the robot will create a new token.

Another problem is what happens any one of the nodes die. This will not allow the token to be passed through the ring, and the network will die. Similarly to the creation problem, a timeout can be introduced, and a new token can either be generated, or the current token can be passed to the next node. The timer will be based on the time since the token was last passed to a node. This should allow for only one token to be created at a time, eliminating multiple tokens.

Nodes will also have to be added to the ring after it is set up and running. When a node is first activated, it enters the timeout phase where it listens for the token being passed. When it hears this, it will listen to find the entire topology of the network. At the end of every pass through the ring, a time is set aside for any new members to enter the ring. The node then announces itself, and is added to the ring.

Each packet sent by the network will have a format that includes a stream of wakeup bits, destination, and some synch data:

Wakeup	Synch	Found	Invite	ECC	Dest ID
31	23	15	14	13	7

Experimental Results

In order to start development of the software on the Boe-Bot, different parts of the Boe-Bot were tested in order to have values for driving the servos, sending the radio frequency data, and using the infrared detectors.

One of the first tasks was to determine correct values to use in pulsing the servos. The servos drive the Boe-Bot's wheels, and are activated by the pulsout command. This command sends a high signal out to the pin for a set period. This period is represented by the number of 2 picosecond time periods in the period. For example a 1 millisecond time period would be put in with a value of 500. This is used to drive the robot forward, backward, left or right. Values for one of the Boe-Bots were as follows:

	Left Servo	Right Servo
Forward	1265	2500
Backward	2535	1300
Right	2500	2535
Left	1000	1000

After testing the servos, we next tested the transmitting and receiving capabilities of the RF devices. With bits of wire for our antennae, we were able to transmit a number, and increment it, and send it back at a range of ten feet. If we limited the BASIC stamp to either transmitting or receiving, it would reach around 25 feet. Further problems came up when the antennae was accidentally touched in some manner.

Design

In order to design the program for running the Boe-Bot, each part of the program will be separated into its various sub programs. This is facilitated by the BASIC 2p chip's ability to poll for changes in values and execute other programs in it's memory when those changes occur. This will allow us to make every program separate, and recreate some of the functionality of structural programming.

The program will consist of 5 parts:

1. An initialization program
2. A program to learn the topology of the network
3. A program to search for the light, and listen for signals in the background
4. A program to receive a token from the previous owner
5. A program to find the node that has the token

Of these, the initialization program has been written and is as follows:

```
node VAR byte
node_next VAR byte
node_list VAR bit(254)
x VAR byte
ID VAR byte
SYNCH CON "A"     'Establish synch byte
BAUD  CON 16780   'N2400 baud (MAX)
trash VAR word
  SERIN 5,BAUD,[WAIT(SYNCH),node]
  SERIN 5,BAUD,[WAIT(SYNCH),trash]
node_list(node) = 1
START:
  SERIN 5,BAUD,[WAIT(SYNCH),node_next]  'get node
  SERIN 5,BAUD,[WAIT(SYNCH),trash]      'trash
node_list(node_next) = 1
  if node = node_next then GET_ID
GOTO START
GET_ID:
  for x = 0 to 254
```

```
    if node_list(x) <> 1 then SET_ID
  next
SET_ID:
  ID = x
'-------wait for Invitation to join ring---------
```

Possible Changes in Design

Further testing may reveal some problems in design, if we cannot pursue our current plans, contingency ones will have to be made. One of these areas in collision detection. The Parallax robotics book that comes with the text implies that the feelers are not as responsive as the infrared detectors for determining objects in the path of the robot. If the feelers cannot give us good enough results, we may have to change to using the IR for detection.

Also, there are some concerns that the infrared may not be strong enough for the robots to hone in on in when a robot finds a light source. We may have to switch to dead reckoning if experiments reveal that the IR does not have the range. We already have a general knowledge of the distances traveled by the Boe-Bot when it goes forward, backward and turns from the pulses sent.

\

Appendix A: Gantt Chart