Maze Solver using iRobots and Wireless Motes

Amee Gosar
a168@umbc.edu

Piyush Waradpande
pi5@umbc.edu

I. INTRODUCTION

Wireless sensor networks have a wide array of use in a number of real life applications. There has been significant amount of research in this field over the last decade, due to its potential of being deployed in applications where human accessibility is limited. These networks find applications in military, disaster management, health care monitoring environment sensing and area monitoring among others. In this project we have developed a wireless sensor network of two iRobots and a base station that solves a given maze. Our project is a first step towards applications like reaching out to victims of natural calamity and rescuing victims of fire accidents without endangering lives of firefighters and rescuers. Such robots also have applications in large warehouses to search for items and deliver them.

We have two modes of operation for our maze solver. The first is a *Discover Phase* in which one node to discovers the structure of the maze. With continuous feedback from the "scout" node, the base station solves the maze. On reaching the destination, the base station understands the maze structure and solves it by. This is the *Solve Phase*. The solve phase generates sequences of commands that can be simply relayed to the iRobot. Upon completion, this solution is forwarded to others node(s) in the network.

In this project we have designed and implemented an easy to use autonomous maze solver. We used IRIS motes and iRobot Create kit for this project, both of which are easily available at reasonable costs.

II. COMPONENTS FOR MAZE SOLVER

A. IRIS Motes

The IRIS motes are wireless sensor modules that are used for wireless communication between the iRobot nodes. We have used the MICAz mote in 2.4 - 2.48 GHz band. The IRIS motes implement IEEE 802.15.4 in 2.4 - 2.48 GHz band for wireless communication. Apart from that, the mote has a 51 pin serial connector that allows serial communication. We programmed the mote using Tiny OS platform in Network Embedded C (nesC) language. A simple python wrapper has been used to provide commands over serial port to the base station. The MIB520 gateway board has a USB connector used to program the motes. For a working demonstration of the project, we used 3 motes, one as a base station connected to the PC and 2 mobile nodes attached to the iRobots. Serial communication between the mote and the iRobot was established using a custom board (provided by course instructor) so that the commands from the IRIS motes can be relayed to the iRobots.

Figure 1: MICAz mote used.

Figure 2: MIB520 USB Gateway

B. iRobot

The iRobot Create is an educational development kit that can perform the common drive tasks of a robot without having to program a custom one. There are two motors to control the

motion of the two wheels. The robot has limited sensors to detect cliff, wheel drop and forward collision. We have used the onboard sensor for forward collision to detect collision. The robot can move in any direction by powering the wheels and giving turn radius. The robot can be programmed via the iRobot Create Open Interface. The OI provides a set of op codes to give commands to the robot. We solve the maze using various combinations of available commands and using an event-driven approach to detect collision.

Figure 3: iRobot Create used as mobile nodes.

C. iRobot- Mote Interface

As mentioned in section II, the interface for communication between iRobot and the IRIS mote over serial channel has been achieved using a custom board. The course instructor provided the board.

III. ALGORITHM

A simple path search approach has been used to solve the maze. Currently we limit the directions of motion towards and bottom and right. But changing the implementation to a depth first search wouldn't be a very difficult task. For purpose of demonstration, we created a maze in the form of a 3x3 grid. Each cell in the grid can be occupied or empty. Empty cells are represented by *1*s and *0*s represent occupied cells. We start with an initially *0*-initialized matrix and scout the iRobot across the grid in a depth first search manner. As we traverse the path, we change *0*s to *1*s for every empty cell encountered. As we encounter obstacles, we change the course of the iRobot and keep going towards our destination.

It should be noted that this scouting requires an event driven approach. We use event commands of the iRobot to achieve information about obstacle location. In the initial *Discovery Phase*, we use the "Bump Event" to calculate number of cells travelled in any direction. A time recording is taken and beginning and after bump event occurs. With known speed and cell-dimension, we find out the number of cells covered. We then mark available cells along the direction as empty.

In the *Solve Phase*, we use our populated matrix to generate a set of commands that can be directly fed to the other iRobot. These commands are generated such that robot travels with a greater speed. Having a greater speed is safe since are collision are totally avoided in the second phase i.e. the *Solver Phase*. This solution is relayed to all the other nodes in the network.

IV. IMPLEMENTATION

We implemented the maze solver using multiple nodes. For demonstration it is recommended to have 3 nodes at least. Node 0 is the base station. One of the nodes becomes the explorer and rest become followers. In our demo with 3 nodes, nodes 1 and 2 are mobile nodes and node 0 acts as the base station. Node 1 is initially an explorer and is used in the *Discovery Phase* to discover the topology of the maze. After communicating the maze structure to the base station, it becomes a follower along with Node 2. Both the nodes now follow commands provided by the base station during the subsequent *Solve Phase.*

We used Tiny OS version 3.0 to program the motes. The programming was carried out in in Network Embedded Systems C and Python. The base station mote is programmed to receive a header through the Serial interface from the PC. The message header specifies which node should the subsequent messages be relayed to. The header is basically a unique byte that does not interfere with the iRobot operations. When the header is missing, the commands are relayed by default to the first mote, making the first iRobot active. The baud rate used for serial communication is 57600 bps. We first send messages to node 1 and carry out the discovery phase and search for the topology.

A python script sends opcodes serially to the base station. The base station relays these commands to the specified node (node 1 in discovery phase) through radio packets. The iris mote receives these radio packets and sends the commands serially to the iRobot. We used a mix of *wait* and *event* opcodes from the open interface for iRobot. For detecting obstacles, we use "Bump" event. As mentioned in section III, we use time-based calculations for recording travelled distance.

In the solve phase, both nodes 1 and 2 (in this scenario where we have a total of 3 nodes in the network) become followers

and accept commands from the base station. The base station sends "optimized commands" to each of the nodes. Optimization is done with two aspects *viz.* obstacle avoidance and speed. It is safe to increase the robot speed since the danger of obstacles is eliminated due to pre-informed safe path.

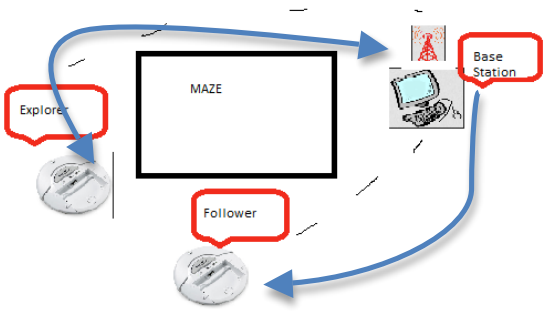

Figure 4: Architecture of the Network.

As suggested by Figure 4, there is a two-way communication between the explorer and the base station and only a one-way communication between the follower and the base station. A two-way communication is needed so that base station can keep track of obstacles encountered by the robot. In the solver phase, a one-way communication is sufficient as the maze is already known and we perform our operations without worrying about hitting obstacles.

V. DEMONSTRATION

We tested the working of our network using 3 nodes. The maze contains a 3x3 grid with randomly placed obstacles in the grid.

A. The Maze

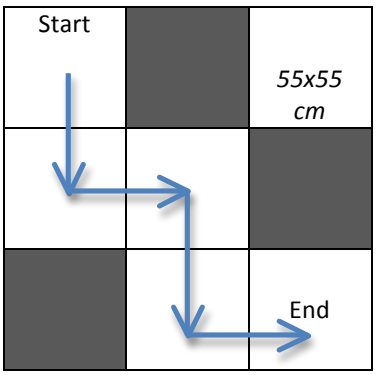

Figure 5: Construction of the Network.

The maze dimension is 165 cm x 165 cm. Each cell in the maze is equally sized at 55 cm x 55 cm. The obstacles in the maze must cover the entire cell i.e. all the 4 boundaries of an occupied cell should be inaccessible. Currently our algorithm does not support having directional accessibility to a cell.

B. Working demonstration

Our base station is connected to a laptop, which communicates to it via serial interface through the python script. The explorer is placed in the cell marked "Start" in figure 4. It finds out a way to get to cell marked as "End". Each move of the explorer is sent to the base station. After receiving all the structure of the maze, the base station solves the maze and finds out the path beginning from start to the end. It generates the sequence of commands to be sent to the follower nodes.

For demonstration purpose, we turn the explorer to a follower and make it solve the maze. Similarly we use another follower iRobot, which traces the entire path dictated by the base station.

Here is the sequence of actions that takes place during the demonstration:

1. We start with 3 nodes: Node 0, 1 and 2
2. Node 0 is connected to a computer and acts as the base station.
3. Node 1 is placed in the "Start" cell facing towards the vertical direction of the "End" cell.
4. On program initiation, Node 1 becomes the explorer and starts scouting through the maze. It moves in two directions, bottom and right.
5. On reaching the "End" cell, the *Explore phase* ends and the *Solve Phase* begins.
6. For purpose of the demo, we instruct first iRobot to pause for duration of 5 seconds. In this duration, we re-position the node 1 to the "Start" cell.
7. The node 1 speeds its way across to the "End" cell.
8. Node 2 (initially placed outside the grid) reaches to the "Start" cell and speeds its way across to the "End" cell.

VI. CONCLUSION

We successfully demonstrated the working of a simple maze solver built using low cost and easily available components. This project introduced us to various kinds of challenges involved in building components involving wireless motes and

their interfacing amongst themselves. We learnt about the basic principles and practices involved in wireless sensor networks and different considerations involved from points of view of energy efficiency, communication efficiency, robustness and durability of a simple network. This hands-on experience with various components introduced us TinyOS platform and programming in it.

The practical usability of such a project can be found with various applications. An example can be automating rescue operations in hostile environments. Similarly, such an explorer-follower model can be used in environments requiring lots of repetitive effort like a mineral ore. Once a mineral reserve is found, multiple worker robots (followers) perform similar operations until the resource is exhausted. Similarly this approach finds its application in automated warehouses where a product has to be found first and then sent carried to another location repetitively.

Some ideas for future scope of this project include the following:

1. Using multiple nodes in the explore phase. This will make the approach more distributed as compared to current implementation.
2. Using proximity sensors instead of on-board collision sensors
3. Using depth first search instead of a basic path finding algorithm
4. Making grid size and cell size generic
5. Reducing communication to base station and performing more tasks on the explorer node

ACKNOWLEDGMENT

We would like to thank Ruthvik Kukkapalli for suggesting us workarounds for problems faced with the iRobot. We would like to thank the TA (teaching assistant) Fahad Alduraibi for helping us out with lots of problems we faced into. We really appreciate that Fahad was extremely helpful during and even outside his office hours. Finally, and most importantly we would like to thank Professor Dr. Mohamed Younis for putting together a fantastic curriculum for CMPE/CMSC 684. We especially appreciate his effort in making the course enjoyable through hands-on experience by making all resources available to us.

BIBLOGRAPHY

[1] Shadi Janasefat, Izzet Senturk,Kemal Akkaya, Micheal Gloff A Mobile Sensor Network Testbed Using iRobots.

[2] iRobot Create Open Interface Manual.

[3]Memsic webstie - http://www.memsic.com/wireless-sensor-networks/