Indoor Navigation using wireless sensor network

Harika Tandra  
Sitapa Rujikietgumjorn  
Neeharika Jarajapu  
University of Tennessee in Knoxville  
htandra, srufikie, njarajap@utk.edu

Abstract

In this paper, we implement a method of indoor navigation on a mobile sensor platform (MSP) using a wireless sensor network. Using tmotes, a network grid of interconnecting nodes is formed for navigating an MSP (Mobile sensor platform) through stationary obstacles in a best path to reach the desired destination. The obstacles are considered to be known a priori. The strength of radio signal of the motes is lowered for communication between the MSP and the tmotes that act as nodes. The MSP navigates to the desired destination in a best path computed by Dijkstra’s algorithm and cross checks itself each time it comes across a grid on its way to make sure that it navigates in the right direction. Incase, the MSP deviates from the path computed, a new best path is computed automatically after identifying the grid it is in. This clearly explains the robustness of our approach. A number of experiments have been conducted on the MSP and the network of tmotes to confirm the validity of this approach.

1. Introduction

Wireless sensor networks are groups of devices which use sensors to measure various types of data such as temperature, light intensity, humidity, pressure and motion at different locations. These sensor devices, called motes, vary from the size of a shoebox to a grain of dust. In addition to one or more sensors, each mote has a radio transceiver, a small microcontroller and an energy source (generally a battery). Their applications include industrial control and monitoring, security and military sensing, healthcare applications, home automation, traffic control, intelligent agriculture and environmental sensing etc.

One growing area of research in wireless sensor networking is its application in indoor navigation. Though a Global Positioning System (GPS) is used for navigation purposes, its performance diminishes greatly indoors. A GPS helps in tracking and navigation by the signals it receives from satellites. As these signals have a poor receptive power in closed spaces than in open areas, their usage for indoor navigation is not efficient and accurate. Therefore, a wireless sensor network can be used as they can be deployed for tracking and navigation purposes in closed spaces.

Cricket motes can be used in determining the position of an object. Though they give accurate performance, they are highly expensive. Hence, we used tmote invents, with some prior assumptions, that give a tradeoff between cost and performance when compared to the cricket motes.

In this project, we form a network of tmotes where each mote represents a small area called grid. Each grid is uniquely identified by a grid number shown on the tmote. The radio signal strength of the tmotes is lowered as they are closely located and also to reduce interference. A base mote is fixed on the MSP which acts as a listener by receiving the packets transmitted from the mote in the grid it is located. The position of the MSP is decided by comparing the strengths of the different radio signals it receives from the grid motes.

Obstacle avoidance is another aspect that is dealt with, in our project. We have considered the obstacles to be stationary and their positions known before hand. Few of the motes are considered to act as obstacles. With the use of Dijkstra’s algorithm, a best possible path is determined for the MSP to navigate from its current position to a specified destination. The path thus computed avoids the grids in which the obstacles are located. Automatic correction of the path is done, if the MSP deviates from the path it is intended to follow and a new path is computed from the grid it is found to reach the destination.

The rest of the paper is divided into the following sections. Section 2 details about the system architecture. In section 3, the technical approach is explained. Section 4 and 5 goes into detail about the experimental setup and results. Conclusions are drawn and future considerations are outlined in Section 6.
2. System Architecture

The goal of the project is to build an indoor navigation system that will guide a robot to move to any given location in that indoor space. This is accomplished by the use of Tmote invents for guiding the navigation and a Mobile Sensor Platform (MSP) which acts as the robot.

The system consists of tmons which are placed on the MSP, and the indoor environment and a computer for initializing the network information in the MSP and starting the navigation process by providing the MSP with location of the final destination. The system block diagram is shown in Figure 1.

2.1 Tmote Invents

The tmons can be divided into two categories based on their use in this project, the grid motes and the decision mote.

Grid mote: The entire indoor area is divided into several grids. Each cell in the grid is a square. The number of cells in the grid and area of each cell is determined by the process explained in section 3.1. Each cell in the grid is uniquely assigned an ID, an integer starting with 0.

A grid mote is placed in the center of each cell of the grid. The grid mote functions as a beacon, they broadcast ID of the respective grid at regular intervals in time. The interval of the broadcast ID depends on the application and the speed of the robot used for navigation; the details are explained in section 3.1.

Decision mote: This mote is placed on the robot. The function of this mote is to keep track of the robot’s location during the navigation. The mote continuously listens to the grid motes and runs a decision algorithm based on the radio signal strength values of the received messages. The output of the decision algorithm is the ID of the grid in which the robot is currently located. The decision mote sends this result to the robot at the start of the navigation and subsequently when the robot moves into a new cell in the grid.

We choose a tmote to compute the decision instead of having the MSP to the job by listening to the grid motes because of the following reasons:

Firstly, the tmons use the AM message type which would have to be converted into the standard type to be understood by the robot.

Secondly, using a decision mote increases the system’s usability as it will work with little change even if any other robot or mobile platform is used in place of the MSP. The only constraint is, it should have an USB port for the decision mote to be hooked.

The grid motes combined with the decision mote provide the indoor coordinate system for this navigation system.

2.2 Mobile sensor platform (MSP)

The Mobile sensor platform is a mobile sensor testbed. The MSP carries devices which enable imaging, distancing and wireless communication functions. They have powerful on-board processing at a low cost of less than $500. The MSP is designed for indoor use and moves around with a speed of about a foot per second.

The MSP consists of VIA EPIA M1000 motherboard running at 1Ghz, a 512MB DDR266 memory and a 20GB hard drive. It uses the 802.11b wireless cards, the Netgear MA311. The MSP moves with help of three wheels. It uses two Dayton gear head motors using 12V DC. The third wheel is the front wheel, which is an omni-directional wheel comprised of eight small rollers. This helps in smooth turning of the MSP. The MSP is installed with Fedora core LINUX operating system. Figure 2 shows the picture of an MSP.
We chose MSP as a robot due to the following reasons. Firstly, the powerful on-board capability of the MSP is well suited for running the complex best path algorithm to reach the destination. Thus, it eliminates the need for a central PC for processing. Secondly, the MSP gives more scope for adding other features to this system in the future, like the detection and avoidance of unknown moving obstacles that can be accomplished using the two sonar sensors and the camera on the MSP. It can also be used for data acquisition and storing purposes in future applications.

In the current system, the MSP computes the best path to a given destination on-board and directs itself along the path with the help of the decision mote and grid motes. The details of how this is accomplished are given in section 3.3.

However there are some limitations to the MSP. Firstly, the MSP does not contain any wheel encoders for reliable movement in straight line. The MSP deviates from the path by an average of 7.5 cm in the lateral direction for 1 meter of travel. Secondly, the MSP can only move on smooth surfaces such as tile etc.

3. Technical Approach

3.1 Network of grid motes

Tmote is a wireless sensor network device that is widely used in an embedded system. As it is a combination between sensor, controller, and radio, it has a capability of sensing the environment through its sensors and communicates with one another through radio communication which can be formed as a network of Tmote. Tmote can communicate with others by broadcasting either to all motes or to a specific mote by indicating the destination ID. Having the capability of forming a network, it extends to wider applications.

![Figure 3. Grid of motes.](image)

In this system, a group of Tmotes is formed as a grid network. These grid motes are used as position points for navigating the robot. Thus, instead of using cricket motes to locate the position, Tmotes can simply be used to locate the position with some considerations such as assuming that the robot has to move in a grid direction. The grid motes will broadcast its ID indicating the grid number as shown in the figure below. This grid number indicates the position of the robot.

As the grid motes are closely placed, there is a need for the radio signal to be lowered, in order to decrease the signal receiving distance to be within the grid and reduce interference. For a Tmote, there are two ways to change the radio signal. The first one is to set the CC2420 radio channel at a compiled time. Another method is to use CC2420Control interface provided by the CC2420RadioC component. The power index can be set in between 1 and 31, where 1 equals to -25dBm and 31 equals to 0dBm (which is the maximum power). On the other hand, the radio signal strength of incoming packet can be read from the RSSI value that is stored in the TOS_Msg structure when it receives a message. It is also possible to get the RSSI value without receiving a message but it is more complicated.

In this project, the power of each grid mote is lowered to 2. One point, worth mentioning, is that the higher the position level of the decision mote from the ground, the better it can receive the signal. That is, to say, the distance to detect a new grid mote will be longer. For this reason, we move the decision mote from the top of the robot to the lower level base of the robot. Once the decision mote receives a message of the grid ID, the received signal strengths will be compared for the stronger signal that indicates which grid is closer to the decision mote.

However, when the receiving mote is at the middle of two grid motes, the result from comparing the strength is not stable. In other words, it keeps showing the results switching between the two motes even though the receiving mote is placed still in between those two motes. This causes some problems as the decision mote needs to get the new direction once it reaches a new grid. So, it will keep getting the switching direction between the two grid motes. Generally speaking, it seems that the accuracy of signal reading is not enough. Thus, threshold is applied to the received signal strength to indicate that the decision mote reaches a specific area within the grid containing the grid mote. The threshold value used for the experiments is 1. Moreover, we let the decision mote wait for a few seconds, once the new grid is found. Then, it informs the robot. The reason for doing so is, if it informs the robot right away, then the point that the robot changes its direction will be too far off from the center of the grid. As a result, it can easily deviate from the grid path. On the other hand, if we decrease this distance, there is a higher chance that the robot will miss the grid when a minor error occurs in the movement. With this delay, there is a wider area for detecting a grid mote and the robot will
change direction only when it is closed to the center of the grid.

Communication between the decision mote and the robot is made via a serial port. Once the robot reaches the new grid mote, the decision mote sends a message, containing grid ID, to the serial port by setting the destination address to be TOS_UART_ADDR. At the start of the process the MSP should signal the decision mote to start listening and get the starting location. This can be implemented by trivial socket programming, however for the current implementation this feature is not available and we manual start the decision mote. In the current system communication between MSP and decision mote is only one way, but the other way can be added easily to the present code. Without this ability the decision mote has to be manually reset when it needs to be start because the decision mote has already got the grid ID and send to the MSP while the MSP hasn’t started yet. So, when the MSP starts, the decision mote will not inform for the new grid because it is still in the same grid which can be temporally fixed by resetting the decision mote.

### 3.2 Dijkstra’s Algorithm: To find the best path

Dijkstra’s algorithm is a graph search algorithm to find the shortest path for a weighted graph. For this project, the graph has a weight equal to one. Below is the psuedocode from Wiki web page [2].

```plaintext
Function Dijkstra(Graph, source):
    for each vertex v in Graph:
        dist[v] := infinity
        previous[v] := undefined
        dist[source] := 0
        Q := copy(Graph)
        while Q is not empty:
            u := extract_min(Q)
            for each neighbor v of u:
                alt := dist[u] + length(u, v)
                if alt < dist[v]
                    dist[v] := alt
                    previous[v] := u
        return previous[]

S := empty sequence
u := target
while defined previous[u]
    insert u at the beginning of S
    u := previous[u]
```

**Figure 4. Psuedocode of Dijkstra’s algorithm**

The grid mote map has to be defined to determine the neighbor and obstacle. In this project, we assume that the obstacles are known beforehand so there is no need to use a sensor to detect an obstacle. As the obstacle position in the map is known, the algorithm can find the shortest path to avoid the obstacle.

Once the path is generated, it must be converted to the direction command values to send to the robot. For example, if the path is 6->3->4, the direction is to go straight, turn right and stop. However, converting the path to direction is not simple since it is relative to the direction the robot is facing. One solution to this is to have a look-up table that tells the direction between two points considering the previous position of the robot. In particular, to go from 4 to 1, there are three possible directions (left, right, or straight), considering from the three previous possible points-- 3 or 5 or 7. This method does not scale up for larger grids.

Hence, we chose to represent the robot’s direction using a simple global direction set consisting of north, south, east, and west. With regard to this information, it is easier to convert the path into directions. In other words, if the current position is north and it needs to be moved to face east, the direction command for this is to turn right. Thus, instead of having a large look-up table, we only need a map that tells the position between two points such as going from point 3 to 4 is east. And with the known current position, the path 3->4 (robot is facing north) is converted into a command for turning right.

However, there is a problem to be concerned with. That is, the first facing direction has to be known and fixed. With this current system of using tmotes there is no way to get the first facing direction. So we assumed that the robot has to first face north, no matter in which grid it is located.

### 3.3 MSP control

The MSP contains two main program modules, the motor control and the serial forwarder module.

The MSP is connected to the decision mote via USB. The messages arriving at the serial port are forwarded to the TCP/IP port using a serial forwarder program (SF). The SF contains two parts, a forwarder which forwards packets from serial port to the TCP/IP socket and a listener program that listens to the TCP port for incoming packets. The listener is programmed to listen in a non-blocking fashion.

The process starts when the MSP receives the grid ID of the destination from the user. The serial forwarder and listener programs are started and the decision mote is signaled to start listening to the grid motes.

On receiving its current grid ID from the decision mote, the best path to the destination is calculated using the algorithm described in section 3.2. The algorithm returns a sequence of directions (“Dir”), the robot has to follow each time it reaches a new grid and also the sequence of grid ID’s of the best path (“Path”).

The MSP then starts the motor control giving the direction parameter. The motor control is designed in
C++. It controls the parallel port to the 2 motors by using simple parallel port programming. The parallel port programming comes with the MSP. We added few functions to this module to set the motor speed and also to turn and move the robot in straight, left, right and back directions.

The motor continues to move in the direction it has been moving until the listener receives a new packet from the decision mote, indicates it’s entry to the next cell in the grid.

The MSP then stops and check the grid id against the next value in ‘Path’ array. If the two match it passes the next direction to the motor control from the ‘Dir’. And the processes repeat. The robot stops when the current grid is matches the destination id.

It can thus be seen that the system is robust and can recover from lost path. This is because the robot checks itself on entering a new cell and if it is lost, it recalculates its path from the current grid to the destination.

The work flow of MSP is shown in Figure 5.

4. Experimental Setup and Results

We tested our system in an indoor space of about (12 x 12) foot. We used 6 grid motes for the experiment. The grid motes are numbered 0- 6. And it is assumed to be placed facing to the north at the start of the navigation. We placed the grid motes on the lab floor, but in case of real applications the grid motes can be hooked to the ceiling of the room. To set the right threshold and radio signal strength (RSS) values for the grid mote. We ran a number of experiments with various combinations of threshold and RSS, the aim was to find the minimum possible cell area into which the area can be divided into. Smaller cell area in the grid will give more accuracy and precision to the navigation. The smaller the area of the cell in a grid the more precise is the coordinate system of the indoor area. In our final setup the cell area was about (3 x 3) foot.

After, the network grid was established and best threshold value was found. The decision mote was connected to the MSP via the USB. For better receiving of the signals the decision mote was placed at the front of the MSP instead of the initially used side position.

On the MSP the serial forwarder and the non-blocking version of serial listener were programmed to listen to the decision mote over the TCP/IP port in a non-blocking fashion. Also the best path algorithm and motor control module was implemented. Use to lack of wheel encoders the MSP shift in the lateral direction on moving some distance in a straight line as mention earlier. To correct this deviation from the straight line path. We have programmed the MSP to stop after moving about 2 feet distance and turn about an angle of 5degrees. Though it slows down the navigation process, it keeps the accuracy of robots movements. Then the MSP was loaded with two arrays. Firstly, a map of the grid, the map contained the adjacency list of each node in the grid. Second array specifies the direction information, it is similar to adjacency list expect that instead of id of the neighbors it gives the direction of its neighbors interns of North, East, West, South.

With this system ready to be tested. We went through several navigation paths by giving different destinations
and placing the MSP at different starting positions. Correct result was obtained with the experiments.

Then we marked some of the areas as obstacles and re-tested the system to get accurate results.

Figures below show some of the screenshots taken while the system is running as viewed from a laptop remotely accessing the MSP.

**Figure 6**

*Figure 6*, shows the best path calculated by MSP when starting cell is 8 and destination is 2. It also shows the code to the corresponding direction it must take on reach next cell in the path. The code is 1: straight, 2: left, 3: right, 4: back, 5: stop. In the above case there is no obstacle. The grid setup for which the above is calculated is shown in figure 3.

Now the same setup is repeated with marking the cell 5 as an obstacle. The results are shown in the screenshot in figure 7.

**Figure 7**

It is seen that the MSP takes an alternate path avoiding the obstacle.

We have come across some challenges while doing the experiments which were not evident in theoretical design of the project. Firstly, the communication through serial port of the motes is not very smooth and is sometimes unpredictable. Second challenge was that the RSS dropped very fast when motes are placed on the floor, so we had to increase the RSS strength.

## 5. Conclusions and future considerations

Through this course project, we tried to present a simple and low cost solution to the indoor navigation problem using the capabilities of the tmotes. Though this system achieves lesser accuracy than the alternate technologies like crickets, it provides a low cost alternative. We have successfully deployed and tested it in the laboratory environment. This system can be further developed to make it more robust for real world applications. We can use the visual and sonar sensors on the MSP to include moving obstacle detection and avoidance. Also, even in its current form the system can be used as a basic navigation tool to build other applications such as data acquisition or area surveillance.

## 6. References


