ABSTRACT

One of the most important applications of wireless sensor networks is building monitoring and more specifically, the early detection of emergency events and the provision of guidance for safe evacuation of the building. In this paper, we describe a demo application that, in the event of a fire inside a monitored building, uses the information from the deployed sensor network in order to find the shortest safest path away from the emergency and provides navigation guidance to the occupants (modelled by a mobile robot), in order to safely evacuate the building. For this demo, we developed our own ad-hoc robot-sensor interconnection using expansion connectors and programming in a low-level language.

Keywords

Emergency management, Building evacuation, Wireless Sensor Network, Robot

1. INTRODUCTION

In the near future, wireless sensor networks will be widely used in smart building monitoring applications. One of the most important applications in this domain is the early detection of emergency events (such as fire or earthquake) and the on-line provision of guidance for safe evacuation of the building. These methods require some form of visual or audio output devices to be deployed within the building to inform users for safe exit paths, e.g. LED signal boards at corridor intersections to direct evacuees towards the safest exit (the nearest exit may not be the safest). Moreover, mobile phones could be used to provide guidance and audio/video information in order to help the evacuees to avoid the hazardous areas in the building.

Furthermore, WSN based emergency systems can support first responders/emergency services to locate occupants still within the building and aid in their evacuation. This is particularly of use in the case of users with restricted movement who may be required to remain in safer meeting points within the building. In [1, 3], the authors propose methods for dynamic evacuation routing using wireless sensor networks.

Our contribution. In this paper, we developed an emergency management application which in the case of an emergency event (e.g. a fire inside the building) uses the information gathered by the WSN in order to find the shortest safest path away from the emergency and provides navigation guidance to the occupants to evacuate the building through the nearest safest exit. The system consists of a WSN that monitors the environmental conditions inside the building, a static base station (Sink) that collects the sensory information and the Evacuation Path application that runs on a desktop PC and provides the calculations for finding the safest path to an exit. Our method is dynamic in the sense that safe exits are re-evaluated continuously to adapt to the way fire expands in the building. For simulation and demonstration purposes, we used a Moway robot that represents the building occupants and we implemented a demo application that simulates the above-mentioned emergency management application.

2. THE BUILDING MODEL

In the current demo, we deployed a wireless sensor network consisting of 12 sensor nodes that monitor the temperature of the building. More specifically we considered one of the floors of a building, where several rooms separated by corridors exist. For demonstration purposes, we used a robot representing the occupants who are guided towards the nearest safest exit (of a total of 3 exits). In order to navigate the robot accurately without large errors in each movement, we tesselated the floor (whose total size is 119cm×119cm) into small (17cm×17cm) squares; each square having specific coordinates (Figure 1) potentially modelling a room of the building.
3. HARDWARE DESCRIPTION

3.1 The Sensor Network

The deployed sensor network consists of 12 TelosB sensor nodes (Figure 2). Berkeley’s TelosB motes (TPR2400) are IEEE 802.15.4 compliant, low powered, open source platforms, with an array of on-board sensors (light, temperature, humidity) [2]. The motes have all been programmed in TinyOS environment [5], a lightweight, low-power platform for wireless embedded systems, using the nesC language. In the current application, each node periodically sends its temperature measurement to the control center (Sink). One of the TelosB motes acts as the base station of the network (Sink) which is connected to a desktop PC and runs the EvacuationPath JAVA application (see Section 4.5).

3.2 The Moway Robot

In order to abstract the building occupants we used the Moway Robot (Figure 3). The Moway robot is a small programmable robot designed to perform different applications, from a simple action to collaborative performance. This robot is equipped with a drive unit which permits it to move in a real environment. The drive unit is connected by a I2C communication bus and together with the sensors to the heart of the robot, a microcontroller PIC. As it incorporates an expansion connector several commercial devices can be connected on it and broaden its applications’ potential.

Our Moway Robot – TelosB Interconnection. We note that the Moway Robot is not equipped with an integrated onboard antenna, thus it is not able to communicate with the Sink. In order to enable the communication between the robot and the rest of the sensor network, we connected a TelosB on the Robot using expansion connectors along with 3 pull-down resistors of 2.2KOhm value (see Figures 4, 5). The 3 General Purpose Outputs of TelosB mote are connected to the 3 General Inputs of the Moway.

4. SOFTWARE ARCHITECTURE

4.1 Sink

The Sink collects and processes the sensors’ measurements. If some temperature reaches a predefined threshold that indicates possible danger, then the evacuation procedure has to be executed: the Sink, using the EvacuationPath JAVA application, calculates the safest shortest path to a building’s exit. The Sink knows the robot’s exact position by exchanging position messages with the TelosB connected on the robot. Using the sensory information, the Sink assigns “safety” weights to the squares of the grid network and
then, using the current robot’s position, calculates the path to the nearest exit, using the Dijkstra shortest path algorithm and sends the new path to the robot.

4.1.1 Implementation in TinyOS

We implemented the following: a) a header file where all necessary structs, message types and macro-variables are defined; b) a configuration file where the wiring among the components used is defined; c) a source file with the source code of the module we implement. We also defined the PathMsg, which is the message type that we will be using for exchanging information in our sensor network.

Each exchanged message includes the current position of the robot, the path sent by the Sink when evacuation needs to be done and the Sensor Network’s temperature measurements. Using the CC2420PacketC component, packets are transmitted when actions need to be performed, i.e. the Sink communicates with the robot only when a new path has to be followed (e.g. in the case of an emergency event or when a random walk is executed by the robot) and receives temperature values when a dangerous threshold is reached.

4.2 The Robot

For enabling the communication between the robot and the TelosB mote, we implemented a program that interfaces their interconnection (see Figure 5). We programmed the General I/O Pins of the robot’s PIC16F876A to be inputs that would be periodically sampled. Thus, when the input state toggles, the robot knows that a different movement command has arrived from the TelosB mote and it has to perform the corresponding movement. Since the robot moves on a grid (Figure 1) there are four possible moves/commands: forward, left, right, rotate (see Table 1).

4.3 The Navigation Implementation in TinyOS

As mentioned in Section 4.1, the mote receives from the Sink a PathMsg message containing a path computed by the JAVA application. Having in mind that the robot can move in four different ways, we have to convert the received path into separate movements. A simple algorithm has been created to automatically compute all the necessary movements upon the reception of a new path message.

Algorithm: The path that the robot has to follow has been calculated by the Dijkstra Algorithm, thus consisting of positions on the grid which have to be translated into simple moves. For example, there are various possible moves for moving from a position (x,y) to another position (x’,y’), depending on the robot’s current direction. If the next position (x’,y’) is in the direction the robot is facing, then a simple forward move must be done. If the next position is on another direction then a calculation must be made in order to find the “must have” direction. The calculations can be seen in Table 2.

4.4 The Sensor Network

The Sensor network is composed by 12 TelosB motes; each of them has been assigned a unique ID. The monitoring application samples every 5 seconds the mote’s temperature using the Temperature/Humidity sensor (Sensirion SHT11) and forwards the data to the Sink.

4.5 The EvacuationPath Java Application

TinyOS provides an interface for communication through UART protocol between the Sink and the Desktop PC. The EvacuationPath JAVA application provides the mechanism for generating safe shortest paths and visualizing the network. It also enables us to initialize the connection between the robot and the Sink.

<table>
<thead>
<tr>
<th>Number</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Movement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Forward</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Left</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Right</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Rotate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Current Position</th>
<th>Next Position</th>
<th>Must Have Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x,y)</td>
<td>(x+1,y)</td>
<td>Down</td>
</tr>
<tr>
<td>(x,y)</td>
<td>(x-1,y)</td>
<td>Up</td>
</tr>
<tr>
<td>(x,y)</td>
<td>(x,y+1)</td>
<td>Right</td>
</tr>
<tr>
<td>(x,y)</td>
<td>(x,y-1)</td>
<td>Left</td>
</tr>
</tbody>
</table>
• Connection Initialization: Firstly, we need to set the Sink in the Listen Mode, so that it can listen messages from the network. The monitoring nodes send periodically their data to the Sink. Also, the robot node sends the robot’s current position to the Sink.

• Dijkstra Algorithm processing: Considering the building’s set-up, we represent the corridors as a grid graph (Figure 1). The building’s set-up can be easily modified as we can freely change the layout of the rooms, the corridors and the exits; thus it can be a generic Grid. The JAVA application, initializes an adjacency matrix which represents the available edges between the corridors and we use it to find the shortest path to an available exit.

In the case of the Emergency Event, the adjacency matrix is updated, making the edges referring to the rooms that have fire unavailable and the Dijkstra Algorithm runs again. The path is sent through the UART connection to the Sink and a packet with the Evacuation Path is being sent through the TelosB radio to the mote connected on the robot.

Moreover, using the adjacency matrix, a Random Walk is being performed by setting a random position in the building as the destination of the robot. When the path is calculated, it is sent by the Sink, through the TelosB radio to the robot. When the robot arrives at the random destination, a new random destination is being calculated and the robot keeps on moving inside the building.

• Data Transfer: The JAVA Application, after calculating the moves for the safe path, sends them to the robot. The connection that has been established between the Sink and the PC, allows us to transfer the data over the radio. For example, after the calculation of the shortest and safest path the application sends a “SendNewPath” request to the Sink, with the path as an argument and the Sink transmits the new path (via the radio) to the robot.

• Visualization: For demonstration purposes, we also developed an online network monitoring application, in order to track the robot’s exact position during the network operation. The JAVA application communicates with the Sink which in turn asks the robot for its position. Then, the Sink receives the current robot’s position and the application visualizes it in a 7×7 GridBox (see Figure 7). The position requests are sent every second in order to always have the updated information about the current position of the robot.

![Figure 6: Moves](image)

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a demo of the Safe Path Evacuation scenario using a wireless sensor network and robot. We presented the hardware components, the software architecture as well as the interfaces implemented. Our application manages to lead an occupant (which is represented by a Moway Robot) out of a building successfully when an emergency event happens, by using on-line updated information from the wireless sensor network in order to find the safest short path to a building’s exit.

We plan to also consider a distributed version of the safe evacuation problem in which the safe path from the evacuee to the nearest safe exit is evaluated on the fly via the local cooperation of sensors.

6. REFERENCES