A Real-time Rescue System: Towards Practical Implementation of Robotic Sensor Network

Jing Yuan* Shao-Jie Tang† Cheng Wang‡ Debraj De§ Xiang-Yang Li† Wen-Zhan Song§ Guihai Chen*
*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P.R. China
‡Department of Computer Science, Tongji University, Shanghai, P.R. China
†Department of Computer Science, Illinois Institute of Technology, Chicago, IL, 60616
§Department of Computer Science, Georgia State University

***Abstract*— A real-time monitor and rescue system must be able to both quickly and reliably detect the event happening in its monitoring region. Furthermore, it is required to fulfill certain rescue mission, *e.g.*, navigate victims to exit through safe path in case of emergency. Current monitor and rescue approaches generally rely on either teleoperated robots, or teams of wireless robots. Typically the robots used in these systems tend to have high cost which make them unpractical in large scale deployment and applications. In this work, we present a real-time monitor and rescue system, TELOSW-BOT NET, utilizing *integrated networks*. The integrated network is an integration of stationary sensor networks and robots: *static sensor networks* comprised of large numbers of small, simple, and inexpensive wireless sensors, and the *robots* which can communicate and controlled by sensor nodes. We demonstrate the efficacy of our system in real test bed composed of 46 sensors, which is one of the largest robotic sensor network to our knowledge, providing empirical results.**

***Index Terms*—wireless network, sensor network, robot, platform.**

I. INTRODUCTION

The recent advances in both the capabilities and miniaturization of wireless sensors have led to applications such as environmental monitoring, event detection and security surveillance. While traditional static sensor networks cannot physically reconfigure themselves to effect more efficient area coverage, in-depth examination of targets, or dynamic protection against inclement environmental developments. By introducing intelligent, robots into traditional stationary sensor networks, it provides us the means to enable sensor networks take an active role in manipulating and interacting with their environment. As pointed out in previous works, simulation was dominating the research methodology in robotic sensor networking. This is because when mobility is added, real-world experimentation often require high labor and equipment costs. In this work, we explore the integration of robots and sensor networks in the context of monitor and rescue applications. We present the design and implementation of TELOSW-BOT NET, evaluate key aspects of its performance, and conduct a few experiments demonstrating its generality.

In specific, TELOSW-BOT NET is implemented through *integrated networks*. The integrated network is composed of static sensor network and mobile sensors: 1) the *static sensor network* contains large numbers of small, simple, and inexpensive wireless sensors, and 2) the mobile sensors can be any robot which has "moving" ability. Broadly speaking, we are interested in the communication and collaboration between robots and the static sensor networks. Essentially, the static network serves as the communication, sensing and computation medium for the robots, and the robots provide actuation, to fulfil rescuing mission, *e.g.*, moving towards the event region and guide victims out of the hazard area. The workflow of TELOSW-BOT NET is illustrated in Fig. 1. The static sensor network is responsible for monitoring and detecting event. When the event is detected, the robot which has "moving" ability will move toward its closest event region, with help of static sensor network, to search victims. After finding the victims, the robot guides them to the closest safe area through safe paths.

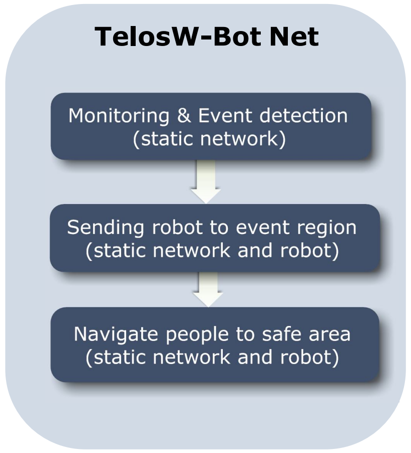

TelosW-Bot Net
Monitoring & Event detection (static network)
Sending robot to event region (static network and robot)
Navigate people to safe area (static network and robot)

Fig. 1. Work flow of TelosW-Bot Net.

We designed and implemented a smart mobile robot, which is equipped with wireless communication unit. In addition, we designed a navigation protocol to implement the application of searching and rescuing. The focus of this paper is on building sustainable and cost efficient robotic sensor network. The main contributions of this paper are summarized as follows:

☐ We design and manufacture a novel sensor mote, called TELOSW. TELOSW nodes compose the static sensor network in our system. It also serves as control unit in the robot. Compared with the traditional sensor node, *e.g.*, TelosB, TELOSW can achieve high event detection accuracy while consuming much less energy.

☐ We design a small and smart robot, called TELOSW-BOT, which is composed of a TELOSW node and motor base. Compared with previous design, it has even lower cost and more reliable performance.

☐ We give practical deployment strategies for static sensor network, which can provide both coverage and connectivity while requiring minimum amount of sensors.

☐ We proposed a GPS-free navigation strategy, focusing on simple algorithms for distributed decision making and information propagation. With help of directional antenna, robot can be guided to any specific static sensor within its communication range (one-hop wide). By employing the concept of potential field, the robot can be navigated anywhere through the static sensor network (network-wide). Different from previous works, we build two kinds of potential fields to accomplish the rescuing mission.

☐ We deploy 46 sensors in the $25m$ by $25m$ parking lot and conduct extensive experiments on real test bed. To our knowledge, it is one of the largest robotic sensor network. Our mobile test bed provides both event detection and path planning. The mobile test bed we have developed plays a clear role in evaluating mobility-related network applications.

The rest of the paper is organized as follows. We present several design consideration in Section II. Hardware architecture are described in Section III. In Section IV, we propose an efficient deployment strategy for static sensors. We describe a two-field based navigation strategy for mobile sensor in Section V. The performance studies of our system are reported in Section VI. We review related works in Section VIII and conclude the paper in Section IX.

II. DESIGN CONSIDERATION

In this section, we present the design considerations behind the development of TELOSW-BOT NET. We also briefly address general issues involved in developing large-scale distributed robotic systems. Specially, the following three requirements together serve as the ultimate goal of our system design.

Practical: To make the system practical and easy to implement, size and cost are two most important factors. For TELOSW-BOT, the dimensions are only $8.0cm \times 5.0cm \times 2.5cm$, and the cost is under \$130. For a single TELOSW node, the cost is around \$100. Compared with most existing systems, *e.g.*, Khepera robot plus radio turret costs approximately \$3000, TELOSW-BOT NET has the lowest cost.

Real-Time: The event detection accuracy and delay are very critical metrics to evaluate the performance of real-time monitoring system. Shorter detection delay and higher detection accuracy often lead to lower damage degree. To improve the detection accuracy, we add a set of new components to TELOSW nodes. As confirmed by our experiments, those new added features provide almost 100% detection accuracy.

Sustainable: To support long term applications, high energy efficiency becomes extremely important as most sensors are battery-powered. However, there is a trade-off between event detection accuracy and energy consumption. For example, by increasing the sensing frequency (consuming more energy per unit time), we can get higher detection accuracy. Again, relying on new features of TELOSW node, we are able to reduce the energy consumption while achieving stratified detection accuracy.

III. HARDWARE ARCHITECTURE

TELOSW-BOT NET is an integration of static sensors and robots (TelosW-Bot):

Static sensor: TelosB node is a possible choice here since it has already been widely used to comprise many sensor networks in various applications. Unfortunately, as we explain later, current design of TelosB node is not sufficient to support the specific requirement of our system. We thus decide to use a newly developed sensor mote, called TELOSW [1], by our group to support our system. Essentially, TELOSW enables TELOSW-BOT NET to be "real-time" and "sustainable";

Robot: The robot used in TELOSW-BOT NET is called TELOSW-BOT. As its name implies, TELOSW-BOT is built from TELOSW node and motor base, with the goal of remaining as simple and flexible as possible. Besides TELOSW node, the motor base is entirely off-the-shelf which makes it commercially available with minimal assembly required. In addition, the software platform for the TELOSW-BOT is based on TinyOS which is a component-based software environment that is designed for deeply embedded systems.

As described above, TELOSW plays a key role in TELOSW-BOT NET. In the following contents, we first introduce the detailed design of TELOSW node. After that, we will describe the design principle, basic structure and functionality of TELOSW-BOT.

A. TelosW Platform

TELOSW is upgraded from TelosB by adding wake-on capability, energy meter and several other external sensors *e.g.*, accelerometer. TELOSW has an integrated design, combining programming, computation, communication and sensing into one device. Users can program on it through standard interface USB connector by using TinyOS. TinyOS is based on the event-driven operating system developed at UC Berkeley for sensor networks. The TinyOS operating system, libraries, and applications are all written in nesC, a new structured component-based language. The nesC language is primarily intended for embedded systems such as sensor networks. The nesC has a C-like syntax, but supports the TinyOS concurrency model, as well as mechanisms for structuring, naming, and linking together software components into robust network embedded systems. The principal goal is to allow application designers to build components that can be easily composed into complete, concurrent systems, and yet perform extensive checking at the compile time.

Due to space limited, we will not go through the detailed design of each component. Instead, we put our focus on the design of wake-on component, which is one of the most important features in TELOSW.

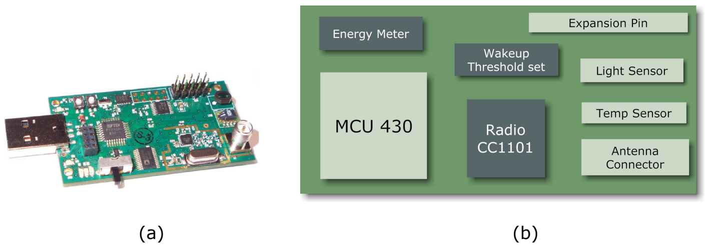


Fig. 2. (a) TelosW Platform; (b) Components of TelosW Platform

In some application, *e.g.*, environmental monitoring, sensor nodes need to collect various environmental data, *e.g.*, temperature, humidity, light intensity, and periodically send data to the sink node through wireless communication. Traditional sensor node, *i.e.* TelosB, works really well for those tasks, where they only need to work with certain duty cycle. However, this kind of work mode is not sufficient to meet the requirements for real-time monitor and rescue system. On one hand, simply letting each sensor node work periodically with high duty cycle will drain out the battery very quickly, which disables the suitability of the system. On the other hand, if we try to reduce the duty cycle to save energy, it will increase the detection latency or even miss the event, which clearly contradicts to the "real-time" requirement. It is worth noting that in monitor and rescue system, the data we really care is those which is "abnormal". Thus, if we are able to design a sensor node which can be waken on by only "abnormal" data, it will significantly reduce the energy consumption while ensuring real-time event detection capability. In addition, by taking account the "practical" requirement, complicated and expensive hardware is strongly discouraged. Instead, we intend to do small modifications on existing sensor node to achieve desired functionality.

To meet all above requirements, we add a set of new hardware components to TELOSW. Benefited from those new components, TELOSW node is able to be waken up in two ways:

Sensor wake-on: It enables the on-board sensors wake up the MCU only on the occurrence of configurable event. This unique design enables the MCU sleep entirely during idle time without missing any event. Detailed hardware design for sensor wake-on component will be discussed later.

Radio wake-on: Wireless communication among different sensors is one of the main functionalities provided by sensor network. However, for traditional sensor node *e.g.*, TelosB, to establish such communication between two sensors, it is required that both sender and receiver stay waken up at the same time. To meet this, we can either let receive always wake up or the CC1101 radio on TELOSW has supporting hardware for WOR, that can save significant amount of energy from communication task. WOR saves energy by using hardware based low power listening and more idle MCU. Besides sensor wake-on, the WOR operation of radio allows MCU to sleep more and wake up only on meaningful events or message reception.

All these event driven sensing and communication design lead to significant reduction in energy consumption. More importantly, as verified in our later experiment, this design ensures highest accuracy and lowest latency for event detection. Detailed description of hardware design for TELOSW can be found in [1].

B. TelosW-Bot Platform

In this section, we introduce the hardware architecture of TELOSW-BOT. TELOSW-BOT contains two main components: TELOSW and *motor base*. The hardware architecture of TelosW-Bot is illustrated in Figure. 3. For TELOSW serving as the static sensor, its hardware design and capabilities have already been described in detailed before. Here we put our focus on its role in TELOSW-BOT.

TELOSW-BOT utilizes TELOSW for its central processing and communication, providing computation and communication capabilities for the robots. It could process sensing data from the sensor boards and control motors through the output signal from ADC. In addition, we add a directional antenna to the TELOSW node which is used for navigation in later stage. The main motivations we choose TELOSW as central controller are summarized as follows:

- Remember that the stationary sensor network is comprised of TELOSW nodes. Using the same mote in the robot enables the effective and reliable interactions between stationary sensor network and the robot;
- New features of TELOSW, *e.g.* sensor wake-on and radio wake-on components, provides the robot the capability of real-time reaction, which is very critical to time-sensitive robots like TELOSW-BOT.

TELOSW-BOT motion is driven by two modified servomotors. The modification"trick" the feedback circuitry so that the servo will stop only when it receives a centering command; it also allows the servo to continuously rotate in either direction. The design of TELOSW-BOT is similar to that of a tank. When both motors are turning in the same direction, the TELOSW-BOT will move in that direction. When two servo motors turn in different directions, the chassis will rotate. The rate of movement or rotation is determined by motor speeds. Each motor is controlled by a pulse-width-modulated (PWM) signal, it allows us to control the direction and velocity of each wheel. The control signal TELOSW sends to the servo's control line is called a "pulse train", TELOSW can be programmed using TinyOS to produce any desired waveform through any of its I/O pins. TELOSW-BOT is powered by two separate power sources: TELOSW is powered by two 1.2 V AA batteries and the motors are powered by four 1.2 V AA batteries.

IV. DEPLOYMENT STRATEGY FOR STATIC SENSORS

A. Coverage and Connectivity

In deploying the static sensors in the monitoring region, two metrics are critical:

Coverage: Each sensor device typically has a physical sensing radius r_s within which it is able to detect the event. An effective deployment strategy should ensure that the entire

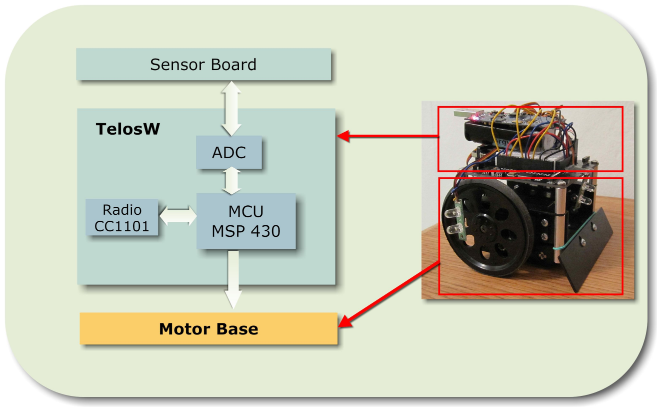


Fig. 3. Architecture of TelosW-Bot

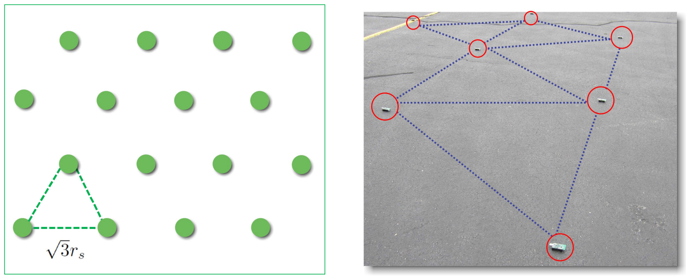


Fig. 4. (a)Triangular lattice structure for deterministic deployment; (b) Part of our test bed.

physical space of interest (or a large fraction of it) is within the sensing range of at least one sensor. The exact sensing range may vary significantly under various applications. In our experiment, we use flashlights as light sources to simulate the event, *e.g.* fire. Thus the sensing region of each sensor can be defined as a disk within which it can detect the shift in light intensity when the flashlight is turned on. Typically, r_s is around $3m$ in our testing.

Connectivity: Each sensor is also associated with maximum physical range r_c for direct communication, which determines its connectivity. Besides the environmental factors, r_c is mainly decided by the transmission power. We set $r_c = 6$ m in our work. As we discuss later, in order to send a robot to the event region and further navigate people to exit, we must ensure that the whole network is connected.

The network topology needs to meet both these requirements of *coverage* and *connectivity*. Any point in the region of interest is provided connected-coverage if it is covered by (*i.e.*, within the sensing range of) at least one sensor node that is connected(*e.g.*, be able to communicate with) to the rest of network. The overall objective of our deployment strategy is to minimize the number of deployed sensors while maintaining connected-coverage.

B. Deployment strategy for stationary sensor network

Assume the monitoring region is a square, we employ the triangular lattice pattern as shown in Figure. 6(b) as our deployment structure. The distance between any two neighbors is $\sqrt{3}r_s$. Note that when the communication radius is sufficient large, *e.g.*, $r_c \geq \sqrt{3}r_s$, this deployment structure can provide both coverage and connectivity. Most importantly, it is optimal in terms of the number of sensors needed when $r_c \geq \sqrt{3}r_s$ [2]. Since we have $r_c = 2r_s$ in this work, the following lemma immediately follows:

Lemma 4.1: Under our experiment setting where $r_c = 6$ m and $r_s = 3$ m, the triangular lattice structure is optimal to ensure both connectivity and coverage in terms of required sensors.

In general, the communication radius r_c and the sensing radius r_s can be different from one another in arbitrary way. It is possible to extend the pattern based topologies discussed above to provide connected-coverage under general setting. For example, when $\sqrt{2} \leq \frac{r_c}{r_s} < \sqrt{3}$, the rhombus-based pattern only requires up to 21% more sensors as compared to the optimal; when $1.14 \leq \frac{r_c}{r_s} < \sqrt{2}$, the square pattern is better up to 60% more sensors than the optimal; when $\frac{r_c}{r_s} < 1.14$, the hexagon pattern requires up to 44% more sensors than the optimal. The detailed proof here is beyond the scope of this paper, interested readers may refer to [3] for more materials.

V. Event Detection & Navigation Strategy

A. Event Detection

In TelosW-Bot Net, the key task of stationary sensor networks is "monitoring", *i.e.*, detect the event once it presents. For different types of events, we must choose appropriate measurements to test its presence. For example, when the event is fire, the readings from light sensor and temperature sensor are important. When the event is earthquake, the readings from acceleration sensor becomes very critical. In this work, we use several Flashlights as light sources to simulate the event, *i.e.*, fire. The presence of event can be detected by the light sensor (Hamamatsu S1087) on the TelosW sensor board. Specifically, when the Flashlights are turned on, the nearby static sensors are able to detect the shift on light intensity. This would cause node to spread alarm by triggering potential field construction. Details about potential field construction will be discussed later.

B. Navigation For TelosW-Bot

After the event is detected and confirmed by the stationary network, we next design a navigation scheme in order to (1) guide the TelosW-Bot to the event region and (2) navigate victims to the closest exit through safe path, with the aid of a directional antenna. The standard TelosW antenna is omnidirectional, but using a directional antenna would allow the robot to determine the moving direction to the target node. Our immediate goal is to achieve the navigation within 1-Hop-wide, that is to guide the TelosW-Bot to a destination node within its communication range. Then we extended our results to network-wide where the destination node could be anywhere in the network.

Fig. 5. 1-hop-wide navigation on test bed.

1) 1-Hop-wide Navigation: We first discuss how to guide the TELOSW-BOT to a destination node within its communication range. In order to fulfill this basic navigation operation, we connect a directional antenna to TELOSW-BOT. And the basic idea is to rotate the directional antenna to measure the Received Signal Strength (RSS) from different directions. Since the directional antenna is fixed on the TELOSW node, we simply rotate robot itself to achieve this. Eventually, it will move towards a direction where the received RSS is maximized. Assume the ID of robot is I_r and the destination node is I_d, we describe the detailed navigation scheme in the following.

1) TELOSW-BOT first broadcasts packet $< I_r, I_d >$ to all nodes within its communication range;
2) Each node within TELOSW-BOT's communication range will be waken up through radio wake-up component. Upon the received message, it first checks whether itself is the destination node (by comparing its ID and I_d). It keeps silent if it is not the destination. Otherwise, it starts to keep broadcasting beacon message $< I_d, I_r >$ until it is no longer the destination;
3) To measure the RSS from different directions, TELOSW-BOT rotates its directional antenna $30°$ every 2 seconds in clockwise direction. Readers may be curious why we set a waiting time of 2 seconds between two rotations. One quick answer is to ensure the accuracy of measured RSS. Notice that when the antenna is rotating, the RSS will become instable which may mislead the guiding direction of the robot. Thus between any two consecutive rotations, we must wait sufficient long time until the RSS becomes stable. As verified by our experiment, setting waiting time to 2 seconds can well balance the length of waiting time and accuracy of measured RSS.
4) After finding out the direction with maximum RSS, TELOSW-BOT moves along that direction until RSS starts to decrease. In this way, it can reach position sufficiently close to destination node eventually.

To this end, we are able to achieve 1-hop-wide navigation successfully. In the following contents, it serves as basic operation in designing network-wide navigation strategy.

2) Network-wide Navigation for rescuing: Once the stationary sensor network detected the event and spread the alarm, the rescuing mission for TELOSW-BOT can be divided into two stages: *searching stage* and *rescuing stage*.

In the searching stage, TELOSW-BOT is desired to move to the event region and get contact with victims; the task of rescuing stage is to navigate victims to the closest exit through safe path. Note that in the first stage, the only objective is "minimum latency". Thus we intend to guide the robot to the hazard area through a shortest path. However, in the rescuing stage, since the robot could be followed by a group of victims, we can not simply guide them to the closest exist through shortest path. Instead we have two requirements with same importance: "minimum latency" and "safety". In specific, the goal is to navigate robot from current location to the closest exit along a curved path, while avoiding any "danger" area. Note that TELOSW-BOT will switch from searching stage to rescuing stage in two possible ways: (1) it can be manually controlled by the victims by through the user button on TELOSW node; (2) it can also be launched automatically after a fixed waiting time (by setting a timer).

The basic idea of our navigation scheme is to build a virtual 3D map, called potential field, in the stationary sensor network, with points of interest (POIs) locating at the "valley bottom". The robot thus "fall" to POI with the help of local information, *e.g.*, the height of the neighboring sensors in potential field. The information at a static sensor that is used to guide the movement of robot is called the potential value of the sensor. When the point of interest is specified (either event region or exit), the node that is closest to the goal triggers the potential field computation. Different from previous works [4], we build two kinds of potential fields: *danger field* and *exit field*. In danger field, the POI is the hazard area, while in the exit field, the closest exit becomes the POI. Each static senor v needs to maintain two potential values: danger potential value $\mathrm{p}_d(v)$ and exit potential value $\mathrm{p}_e(v)$. In searching stage, TELOSW-BOT utilize danger field to get close to the hazard area. The exit field will be utilized at the rescuing stage to guide TELOSW-BOT to the closet exit through safe path.

`Danger Field Construction`: We adopt a simple Breath First Search (BFS) Tree based method to calculate the danger potential value $\mathrm{p}_d(v)$ for each static sensor v. Initially, all the static sensors v_i set $\mathrm{p}_d(v_i) = n$. Without detecting any event, the initial potential field is flat. Once a static sensor v_d detects the event, it sets $\mathrm{p}_d(v_d) = 0$ and triggers the danger field computation immediately. The danger field is built in a pure distributed way as shown in Algorithm 3. In this way, the danger potential value of each static sensor becomes its hop distance from its closest POI. Clearly, the POI has the minimum potential value and a non-POI sensor closer to POI obtains lower potential value. Guided with static sensors within its communication range, TELOSW-BOT always move toward a sensor of lowest potential and eventually reach the target. This can be achieved by 1-hop-wide navigation strategy proposed previously.

Lemma 5.1: By following danger field, TELOSW-BOT can reach the event region through a path with minimum hop distance in the communication graph.

`Exit Field Construction`: Compared with the danger field, it is more complicated to build up the exit field. Remember that the task of the rescuing stage is to navigate robot to the closest exit while avoiding any danger area. Thus when constructing the exit field, we must take into account any

Algorithm 1 Danger Field Built-Up for each v_i

```
1:  $\mathrm{p}_d(v_i) = n$;
2:  if $v_i$ detects and confirms a event then
3:      $\mathrm{p}_d(v_i) = 0$;
4:      Broadcast updating message containing $\mathrm{p}_d(v_i)$ to its neighbors;
5:  end if
6:  if $v_i$ receives a updating message from $v_j$ then
7:      if $\mathrm{p}(v_i) > \mathrm{p}(v_j) + 1$ then
8:          $\mathrm{p}(v_i) = \mathrm{p}(v_j) + 1$
9:          Broadcast updating message containing $\mathrm{p}_d(v_i)$ to its neighbors;
10:     end if
11: end if
```

Algorithm 2 Exit Field Built-Up for each v_i

```
1:  Initial State:
2:  $\mathrm{p}_e(v_i) = n$;
3:  if $v_i$ is the closet sensor to some exit then
4:      $\mathrm{p}_e(v_i) = 0$;
5:      Broadcast updating message containing $\mathrm{p}_e(v_i)$ to its neighbors;
6:  end if
7:  if $v_i$ receives a updating message from $v_j$ then
8:      if $\mathrm{p}_e(v_i) > \mathrm{p}_e(v_j) + 1$ then
9:          $\mathrm{p}_e(v_i) = \mathrm{p}_e(v_j) + 1$;
10:         Broadcast updating message containing $\mathrm{p}_e(v_i)$ to its neighbors;
11:     end if
12: end if
13: Emergency State:
14: if $v_i$ detects a event then
15:     $\mathrm{p}_e(v_i) = n + \mathrm{p}_e(v_i)$;
16:     Broadcast updating message containing $\mathrm{p}_d(v_i)$ to its neighbors;
17: end if
18: if $v_i$ receives a updating message from its neighbor then
19:     Find the minimum exit potential value $z$ among all its neighbors;
20:     if $(\mathrm{p}_e(v_i) \neq z + 1)$ and $(\mathrm{p}_e(v_i) \neq 0)$ then
21:         $\mathrm{p}(v_i) = z + 1$;
22:         Broadcast updating message containing $\mathrm{p}_d(v_i)$ to its neighbors;
23:     end if
24: end if
```

Algorithm 3 Motion Strategy for TelosW-Bot

```
1:  TelosW-Bot periodically checks the potential value of each static sensor in its communication range;
2:  if there exists a static sensor $v_j$ with lowest potential value then
3:      Move toward $v_j$ using 1-hop-wide navigation strategy;
4:  end if
```

possible hazard area that may exist on the navigation path. To meet both "minimum latency" and "safety" requirement, we divide the construction into two parts: *initial state* and *emergency state*.

Assume no event happens at the initial state, we first construct a exit field with the nodes that are closest to some exit as POI. Once some event is detected, we construct a "virtual wall" around the hazard region by forcing those sensors within hazard area increase their exit potential value. By adaptively adjusting current exit potential value of each senor, we can successfully guide the robot to the closet exit through shortest path while avoiding those "wall"s.

In the initial state where no event happens, we apply the similar scheme used in danger field construction to build up the exit field. The only difference is that we set the sensor which is closest to some exit as POI. This initial construction ensures that TELOSW-BOT can move toward its closest exit correctly when no event happens. Once some static sensor detects the event, it triggers the emergency state in which each sensor v will adaptively modify its current exit potential value as follows:

- When sensor v detects the event, it sets $\mathrm{p}_e(v) = n + \mathrm{p}_e(v)$ and broadcasts its current potential value to its neighbors;
- When sensor v receives updating message from any of its neighbor, it will wake up all its neighbors through radio wake on and compute current minimum exit potential value z among them. After z is found, v compares its original exit potential value $\mathrm{p}_e(v)$ with $z + 1$, if $\mathrm{p}_e(v) < z + 1$, v sets $\mathrm{p}_e(v) = z + 1$ and broadcasts updating message to its neighbors.

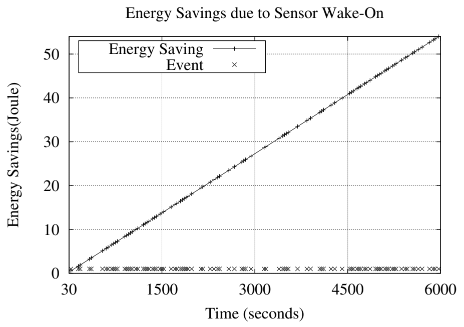
In the first case, we force those sensors within hazard area push themselves to most-top of the exit field. To achieve this, we add n to their original value. There are two main reasons here: 1. the robot intends to avoid those hazard areas as the potential values of all those sensors are sufficiently high; 2. Remember that the original exit potential value (calculated in initial state) reflects the hop-distance to the closet exit. By adding the same number n to their original value, we can ensure that even within the event region, the robot can still move towards the closest exit through shortest path.

In the second case, $\mathrm{p}_e(v) \neq z + 1$ implies that the original shortest path from v to the exit has been destroyed. For example, some node in the previous shortest path falls into the hazard area. The system will perform local adjustment of $\mathrm{p}_e(v)$ to compute a new shortest path by avoiding those "danger" nodes. It is worth noting that not all sensors need to adjust their exit potential value, for example, for those sensors whose original shortest path is not affected will not do any adjustment.

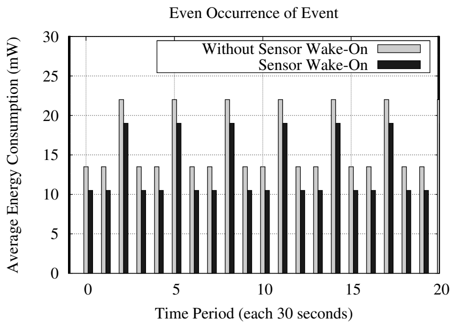
To this end, we have finished the construction of two potential fields. The motion strategy for TELOSW-BOT is same on both fields.

VI. EVALUATION I: TELOSW NODE

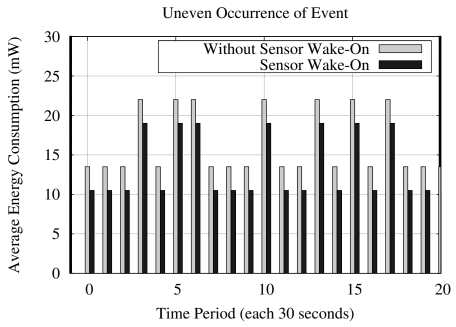
In this section, we evaluate system performance of a single TELOSW node in terms of energy efficiency and event detec-



(a)



(b)



(c)

Fig. 6. Energy savings due to wake-on sensor.

tion accuracy. As energy efficiency is a important measurement of sustainability of the system, we first conduct several experiments to evaluate the energy consumption of TELOSW node. On the other hand, the detection accuracy is a very critical metric to evaluate system performance. For the events such as fire, a shorter detection delay significantly reduces the amount of damage. The following experimental results confirm that the TELOSW node can meet pretty high detection accuracy(almost 100%) while consuming less energy.

A. Energy Efficiency from Sensor Wake-On

To reduce the energy consumption and ensure the sustainability of our system, we employ sensor wake-on capability to TELOSW node. The wake-on sensor capability lets the MCU sleep, while the sensors (light, accelerometer, external sensor) trigger interrupt to wake MCU up. Therefore the MCU does not have to sample the ADC until interrupted by only meaningful events. In this section we have done experiments in order to assess and validate the savings of energy due to wake-on sensor. All the results support the capability of energy efficient wake-on property of TELOSW.

The experiments are set up as follows. Two TELOSW nodes are programmed in two modes: (i) say node 1 without sensor wake-on, and (ii) say node 2 with sensor wake-on. Node 1 periodically samples the light sensor reading, while node 2 uses sensor wake-on for light sensor (so that it will be triggered only if light reading is above 10). Both of them send serial message if any light event (light intensity above 10) is detected. Additionally both the nodes send energymeter reading periodically through serial message. Now during some of the 30 seconds periods, one light event is generated. First the light event is generated with uniform interval(Figure. 6(b)), then generated with non-uniform interval (Figure. 6(c)). It can be observed that node 1 (without sensor wake-on) consumes more energy than node 2 (with sensor wake-on) during all the period. In each period, node 2 saves around 3 mW more energy than node 1. The sensor wake-on capability of TELOSW provides this advantage from idle MCU.

The second experiment is conducted to measure the energy savings in long term. Through 6000 seconds, 94 light events (light intensity above 10) are randomly generated. As illustrated in Figure. 6(a), the sensor wake-on mode saves about 54 Joule in 6000 seconds. This shows that for long term real-time applications, the sensor wake-on capability provides significant energy savings and therefore much longer lifetime.

B. Event Detection Accuracy:

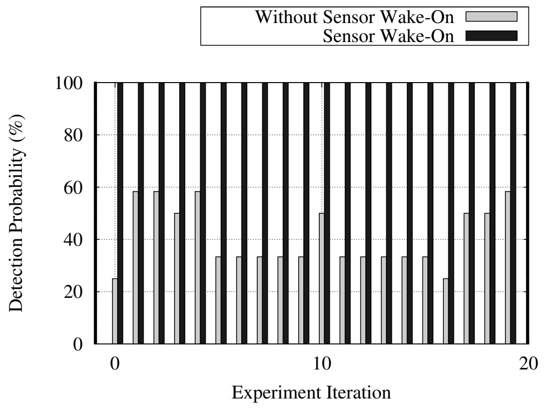


Fig. 7. Light event detection accuracy under same energy consumption.

Without sensor wake-on mode, we still achieve low energy consumption by reducing duty cycle. However, this comes with less frequent sensing, thus may miss a number of events from detection. We next show that with help of sensor wake-on, TELOSW can achieve almost 100% detection accuracy while consuming pretty low energy. As shown in Figure. 7, through 240 periods (each period 30 seconds long), the node without sensor wake-on sense for 3 seconds every 10 seconds (with duty cycle around 33%). The light event is generated randomly within each 30 seconds period. Then from averaging the performance, it can be observed that with sensor wake-on, the node does not miss any event. This is because the configured sensor wake-on circuitry wakes MCU only when the event happens. But without sensor wake-on the node detects the event only with probability between 33% to 58%. This is because for all varying period of sensor reading, the MCU is not always reading the sensor exactly during the occurrence of the events. On the other hand, the energy consumptions of these two modes are almost the same(around

40 mW). Therefore the sensor wake-on capability of TELOSW provides almost 100% event detection accuracy with less energy consumption from more idle MCU.

VII. EVALUATION II: IMPLEMENTATION OF TELOSW-BOT NET

In this section, we describe the results of experiments using our mobile test bed. We deployed 46 TELOSW nodes and one TELOSW-BOT in a $35m$ by $35m$ parking lot. These 46 sensors are deployed in triangular lattice structure as explained in Section IV. They are serving as stationary sensor network in the following experiments.

To simulate the events, *i.e.*, fire, we utilize several Flashlights. When the Flashlights are turned on, the nearby static sensors are able to detect the shift on light intensity (based on the readings from light sensor). Thus the presence of event can be detected by those sensors. In our experiment (outdoor environment), the light event can always be detected by the sensors less than $3m$ from the LED. Thus the sensing radius can be roughly set to $3m$. Based on the discussions from Section IV, we can determine the distance between neighbors as $\sqrt{3} \cdot 3 \approx 5m$. After the event is detected, the static sensor network will spread alarm and trigger the potential field construction. Detailed description can be found in SectionV. We are interested at the potential field construction as well as the moving trajectory of the robot on real test bed.

We next conduct two experiments to test the performance of our system. In the first experiment, we turned on five Flashlights in the network. Figure. 8(a) depicts the constructed danger field. The map is bilinearly interpolated from the danger potential value, with the nearest two sensors contributing to the points in the map between them. There are totally 10 sensors detecting the event and two disjoint event regions have formed. Figure. 8(b) illustrates the resulted exit field with exit located at (31,0). As we expected, the sensors located within the event region push themselves to most-top of the exit field. The sensors whose original shortest path have been destroyed also adaptively adjust their potential value. Figure. 8(c) shows the moving trajectory of TelosW-Bot. Guided by the danger field, TelosW-Bot first moves toward its closest event region (searching stage). After staying at the event region for 10 seconds, it switches to rescuing stage automatically. Note that this switch can also be controlled manually by the victims, in this experiment, we simply set a timer to achieve this for simplicity. As confirmed by the experimental results, in the rescuing state, TelosW-Bot moves towards the closet exit while avoiding the danger area. We repeated this experiment immediately after the first run had completed, by change the event region, initial position of robot and exit position. Similar information is shown in Figure. 8(d)(e)(f). The overall results show that our system performs good in real implementation.

VIII. RELATED WORK

Some research efforts have been carried out on the implementation of mobile sensor nodes. Three of the most famous experiments were Robomote [5], MICAbot [6], and CotsBots [7]. All of them used the Motes [8] series of products as their central processing control and communication. However, all above experiments did not really integrate the robot with the sensor network but only focus on the robot design. Researchers in robotics have also discussed the surveillance issue [9]. Robots or cameras installed on walls identify obstacles or humans in the environment. These systems guide robots around these obstacles. Such systems normally must extract meaningful information from massive visual data, which requires significant computation or manpower. Navigation is a fundamental problem in mobile robotics. A number of solutions [10] [11] have been proposed to resolve this problem. All of the approaches have assumed, however, that a map of the environment was available in advance. [12] used wallboard cameras to capture mobile sensors' locations.

IX. CONCLUSION

We have described the design and implementation of a robotic sensor network. Our experimental results so far shows it to be a promising testbed, valuable for a range of experiments in mobile and wireless networking. Obviously, much more may be accomplished using the TELOSW-BOT Net. Work has already begun in deploying multiple robots. In this case, the robots need to communicate and share information with each other to achieve better performance. We are also working on deploying and testing our system in indoor environment.

X. ACKNOWLEDGEMENT

The research of authors is partially supported by the National Basic Research Program of China (973 Program) under grants No. 2010CB328101, No. 2011CB302804, and No. 2010CB334707, the Program for Changjiang Scholars and Innovative Research Team in University, the Shanghai Key Basic Research Project under grant No. 10DJ1400300, the NSF CNS-0832120, NSF CNS-1035894, the National Natural Science Foundation of China under grant No. 60828003 and No. 61003277, the Program for Zhejiang Provincial Key Innovative Research Team, and the Program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), China NSF grants (60825205, 61073152).

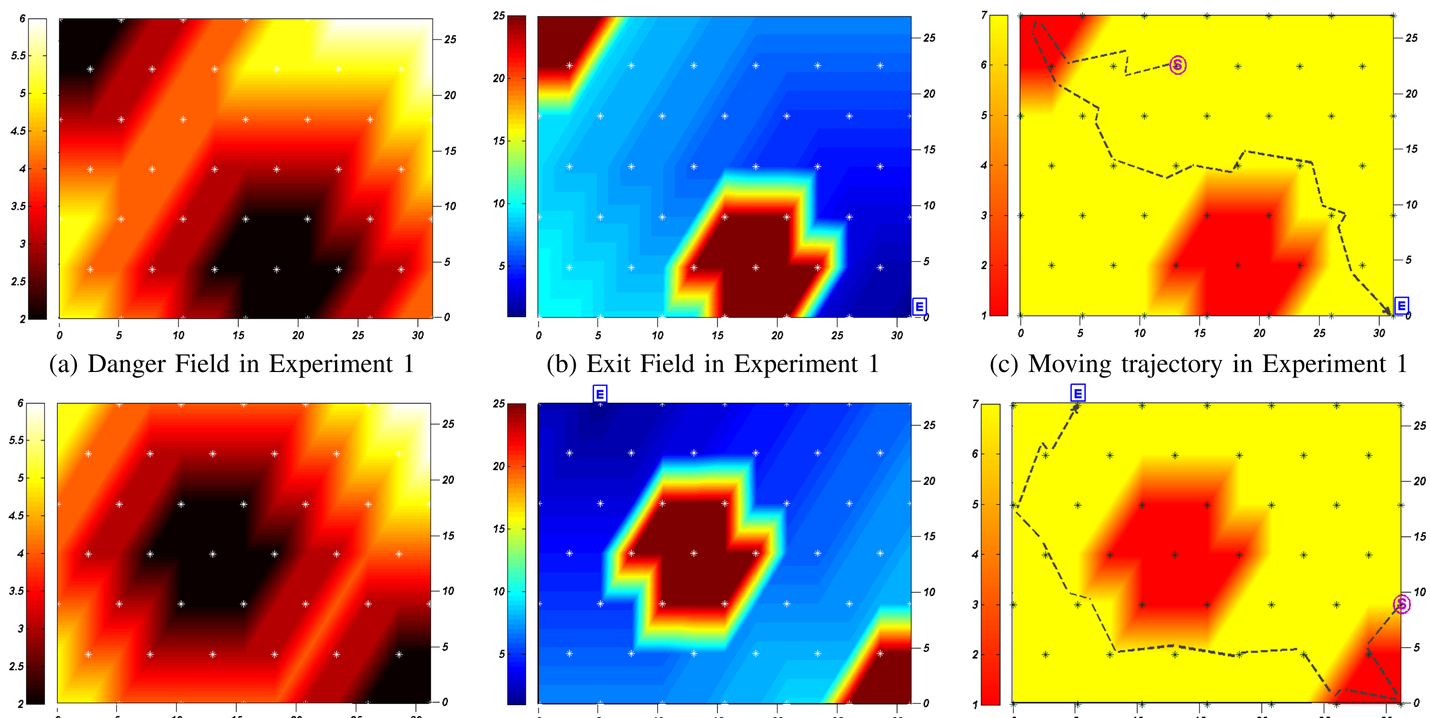
REFERENCES

[1] G. Lu, D. De, M. Xu, W. Song, and J. Cao, "TelosW: Enabling ultra-low power wake-on sensor network," in *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*. IEEE, pp. 211–218.

[2] R. Iyengar, K. Kar, and S. Banerjee, "Low-coordination topologies for redundancy in sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2005, p. 342.

[3] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. Lai, "Deploying wireless sensors to achieve both coverage and connectivity," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2006, p. 142.

[4] Q. Li, M. De Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003, p. 325.



(a) Danger Field in Experiment 1 (b) Exit Field in Experiment 1 (c) Moving trajectory in Experiment 1

(d) Danger Field in Experiment 2 (e) Exit Field in Experiment 2 (f) Moving trajectory in Experiment 2

Fig. 8. Danger field, exit field and moving trajectory in two experiments.

[5] G. Sibley, M. Rahimi, and G. Sukhatme, “Robomote: A tiny mobile robot platform for large-scale ad-hoc sensor networks,” in *Proceedings-IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1143–1148.

[6] M. McMickell, B. Goodwine, and L. Montestruque, “MICAbot: a robotic platform for large-scale distributed robotics,” in *IEEE International Conference on Robotics and Automation*, vol. 2. Citeseer, 2003, pp. 1600–1605.

[7] S. Bergbreiter, K. Pister, B. Sensor, and C. Actuator Center, “Cotsbots: An off-the-shelf platform for distributed robotics,” in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings*, vol. 2, 2003.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System architecture directions for networked sensors,” *ACM Sigplan Notices*, vol. 35, no. 11, p. 104, 2000.

[9] J. Lee and H. Hashimoto, “Controlling mobile robots in distributed intelligent sensor network,” *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 890–902, 2003.

[10] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” in *IEEE International Conference on Robotics and Automation*. Citeseer, 1999, pp. 1322–1328.

[11] K. Arras, N. Tomatis, B. Jensen, and R. Siegwart, “Multisensor on-the-fly localization::: Precision and reliability for applications,” *Robotics and Autonomous Systems*, vol. 34, no. 2-3, pp. 131–143, 2001.

[12] D. Johnson, T. Stack, R. Fish, D. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, “Mobile emulab: A robotic wireless and sensor network testbed,” in *IEEE INFOCOM*, 2006, pp. 23–29.