CotsBots: An Off-the-Shelf Platform for Distributed Robotics

Sarah Bergbreiter and K.S.J. Pister
Berkeley Sensor and Actuator Center (BSAC)
497 Cory Hall
Berkeley, CA 94720 USA
sbergbre,pister@eecs.berkeley.edu

ABSTRACT

The CotsBots are inexpensive and modular mobile robots built entirely from commercial off-the-shelf components. These robots provide a convenient platform on which to investigate algorithms, cooperation, and distributed sensing in large (> 50) robot networks. Each robot is small (13cm x 6.5cm base) and costs under $200. Each is equipped with on-board processing, radio communication, and a base platform for mobility. Software is written using TinyOS, an open-source, event-driven operating system for large-scale distributed sensor and actuator networks. TinyOS also provides a modular software environment where implementation details may be abstracted away from the robot application developer. A simple robot diffusion algorithm has been outlined to demonstrate the ease of using CotsBots for large-scale systems.

Keywords

Multi-robot Systems, Distributed Robotics, Mobile Robots, off-the-shelf, Sensor Networks

1. INTRODUCTION

Until recently, research in multi-robot systems has been limited due to the size, cost, and complexity of the various robots used. Khepera robots seem ideal for their small size and off-the-shelf hardware platform, but each Khepera robot plus radio turret costs approximately $3000 [1]. Pioneer robots common throughout multi-robot research today prove to be prohibitively large for a small lab environment [2]. Other newer robot platforms such as the Robomote, MICAbot, and the Millibots provide small, economical platforms for developing mobile robot applications, but suffer from the custom design approach historically associated with the robotics community [3,4,5].

In addition, programming multi-robot systems has also been extremely cumbersome due to programming languages that do not readily support networked systems. The application developer is often required to write lower level drivers and a new network stack for each different robot used. Instead, a multi-robot research platform should put the emphasis on application development, yet provide enough flexibility to modify lower level software as necessary.

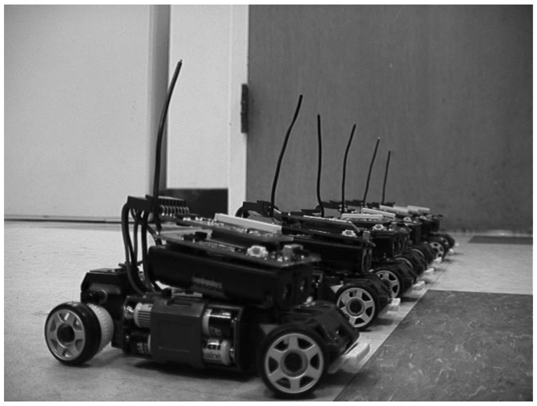
Figure 1. A line of CotsBots. Over 80 robots have been built to date.

This paper presents an inexpensive and modular robot platform built entirely from off-the-shelf components (CotsBots). Each robot is small (13cm x 6.5cm base), costs under $200, and is equipped with on-board processing and radio communication. Each of the pieces comprising the CotsBots are commercially available with minimal assembly required (off-the-shelf). In addition, the software platform for the CotsBots is based on TinyOS, an open-source, event-driven operating system for large scale sensor and actuator networks. The CotsBots satisfy all of the requirements for a multi-robot research platform. They are small, cheap, off-the-shelf, and easily programmed (Figure 1).

2. DESIGN CONSIDERATIONS

Design considerations in this space have already been discussed in regards to the Robomote and MICAbot mentioned above [3,4]. Size and cost are important factors when designing a large-scale multi-robot platform since research should be feasible within a reasonably sized lab space and budget. However, the basic functionality required by a robot platform such as the CotsBots is still an open question.

The CotsBots are intended as a platform for multi-robot research. Therefore, flexibility is essential in the platform to enable a variety of research directions. For

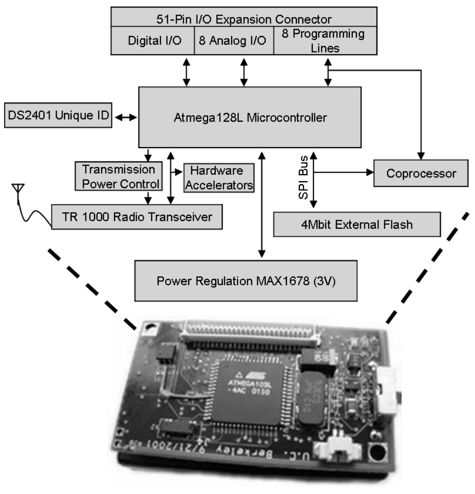
51-Pin I/O Expansion Connector
Digital I/O
8 Analog I/O
8 Programming Lines
DS2401 Unique ID
Atmega128L Microcontroller
Transmission Power Control
Hardware Accelerators
SPI Bus
Coprocessor
TR 1000 Radio Transceiver
4Mbit External Flash
Power Regulation MAX1678 (3V)

Figure 2. The Mica Mote hardware architecture. The Mica provides processing and communication as part of the base functionality of the CotsBots (courtesy J. Hill).

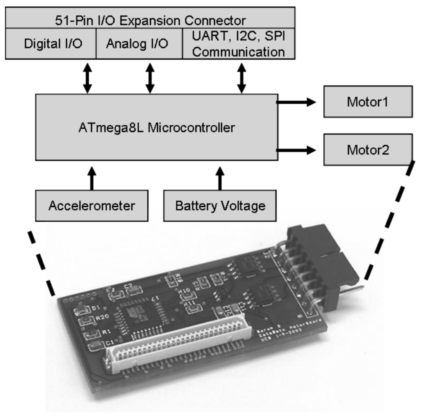

Figure 3. The MotorBoard hardware architecture. The MotorBoard provides actuation for the CotsBots.

example, a mapping application might require sonar sensors to detect distances to objects as well as sensors for localization. A simple group behavior algorithm may only require a radio.

It was therefore determined that the base functionality of the CotsBots should provide only processing, communication, motor control and power: the absolute necessities in a multi-robot application. It is hoped that sensors will be added as required by the application. Section 3 will discuss available sensors and sensor possibilities further. Flexibility is further addressed through both hardware and software modularity as discussed in Sections 3 and 4 below.

It is also important to note that the CotsBots have been developed within the context of large-scale sensor networks. Yet another reason to keep the base functionality on the CotsBots as simple as possible is that sensing functionality may be ported to a surrounding network of sensors instead. For example, localization and tracking information may be provided to the robot from external sensors throughout the area in which the robots are travelling [6].

3. HARDWARE ARCHITECTURE

The CotsBots are built entirely from off-the-shelf components with the goal of remaining as simple and flexible as possible. Four main components comprise the standard CotsBots hardware configuration: the Mica mote, MotorBoard, sensors, and base.

3.1 Mica Mote

The Mica mote (Mica) is responsible for communication, most of the processing, and most of the sensor interfaces. The Mica is a commercially available platform originally built for large-scale distributed sensor networks [7] and is a more recent version of the hardware platform described in [8]. It is based on the ATmega 128L processor, a low-power AVR 8-bit processor with 128 Kbytes of flash program memory, 4 Kbytes of EEPROM and 4 Kbytes internal SRAM. The ATmega 128L also includes an 8-channel 10-bit ADC, three timers, and several bus interfaces including SPI, I2C, and two USARTs.

One of the unique features of the Mica mote is the integrated 916MHz RFM radio used for communication. Maximum speed is currently set to 40kbps and ranges from 10ft up to 100ft have been common. The radio may also be used to program the robots. When using more than fifty robots for an application, network programming is essential for the sanity of the programmer.

Furthermore, a 51-pin connector interface allows the connection of additional sensor and actuator boards as described in Sections 3.2 and 3.3. A block diagram of the Mica mote architecture is shown in Figure 2.

3.2 MotorBoard

The MotorBoard is responsible for actuation on the CotsBots. This board utilizes its own processor, the ATmega 8L, to provide maximum flexibility in interfacing with the other CotsBots components. The ATmega 8L provides 8 Kbytes of flash program memory, 512 bytes EEPROM and 1 Kbyte internal SRAM as well as an integrated 10-bit ADC, several timers with PWM support, and the SPI, I2C, and UART bus interfaces.

In addition to the ATmega 8L, the MotorBoard provides an interface to two discrete MOSFET H-bridge circuits, which can be used to provide speed and direction control for motors, positioning for servos, switching relays or solenoids and a variety of other actuation schemes. Each H-bridge circuit is specified to handle over 4A at 30V, although this will be considerably less without the use of an external heat sink.

The MotorBoard utilizes the same 51-pin connector mentioned earlier to communicate with the Mica mote over UART, I2C or SPI protocols. To attach the MotorBoard to the robot base, a standard 0.1” Molex 8-pin connector is used.

The MotorBoard also provides a circuit for an Analog Devices ADXL202e 2-axis +/- 2g accelerometer. The accelerometer is an optional component on the MotorBoard and not considered to be part of the CotsBots’ base functionality. However, it may be useful for collision detection or very crude inertial navigation where input directly to the motor controller is desired. The MotorBoard hardware architecture is shown in Figure 3 and schematic/layout files are available from [9].

3.3 Sensing

The Mica motes were originally designed for large-scale sensor networks, and therefore, a variety of sensor boards are commercially available to interface with the Mica. Optional sensor boards include light and temperature sensing, a Honeywell HMC1002 2-axis magnetometer, an Analog Devices ADXL202e 2-axis, +/- 2g accelerometer, a 4kHz buzzer and a microphone. Another sensor board developed for the Mica includes a humidity sensor, pressure, thermopile, temperature and light. This board was used to detect the presence of nesting birds as described in [10].

A small sampling of robot functions achievable using these sensors include collision detection using the accelerometer, detecting human presence, compassing with the magnetometer, and very crude time-of-flight measurements using the radio and buzzer/microphone pair [13]. Additional sensing capability may be added through integration with the 51-pin connector.

3.4 Base

The base platform for the CotsBots supplies mobility as well as power to drive the robots. While any off-the-shelf base may be used, most of the CotsBots are currently built from a Kyosho Mini-Z Racer RC Car [11]. By using a toy car, the motors, wheels, and batteries are already conveniently packaged and easy to use. Another example of a CotsBots base is the Plantraco Desktop Rover, a differential drive platform [12]. Software is very easily modified to take this into account as described in Section 4

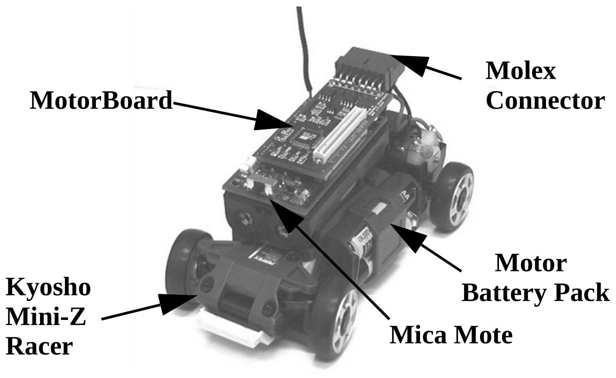

Figure 4. Close-up of the CotsBots platform. Every component is off-the-shelf and minimal assembly is required to connect the pieces.

Using the Kyosho Mini-Z Racer RC Car platform, power is provided by 4 AAA batteries and each robot can drive continuously for approximately one hour. The Mica is sold with an attached battery pack which is used to power the digital circuitry. Using no power conservation algorithms, a typical application on the Mica lasts just under one week.

Very few modifications are required to connect the robot base to the rest of the hardware platform. Any original electronics on the RC car platform are removed, and motor/servo/solenoid wires are connected to a standard 0.1” Molex 8-pin connector which plugs into the MotorBoard described above (Figure 4).

4. SOFTWARE ARCHITECTURE

Software is based on the event-driven TinyOS operating system developed at UC Berkeley for sensor networks [8]. In an event-driven operating system, software responds to events that are propagated from the interrupt level to higher, more abstract system levels. For example, a beep heard on the microphone signals a hardware interrupt which further signals a beepDetect() event at the application level. The event-driven TinyOS architecture may also be used to implement simple finite state machines as seen in [4].

TinyOS offers several other advantages as well, including the ability to abstract software modules. Much of the software described below is written so that the application developer does not need to know what hardware is used or how to interact with that hardware. Instead, abstracted functionality such as setSpeed(), setDir() and setTurn() commands for robot control are provided to promote software reuse and to simplify the software coding process.

TinyOS is an open-source, component-based software platform. This is of great advantage in the re-use of software. Many components, such as the network stack and lower level driver components, have already been written. CotsBots programmers are therefore free to focus on robot-related components and plugging these components together to create applications. Both TinyOS and CotsBots software is available in the Sourceforge Repository [9].

Currently, software for the CotsBots is divided into two parts: software that runs on the MotorBoard and software that runs on the Mica Mote. The two boards interact through the 51-pin data bus using either UART, I2C or SPI protocols. Figure 5 is useful for visualizing this interaction.

4.1 MotorBoard Software

MotorBoard software runs on the ATmega8L and consists primarily of lower-level components required to drive the motors. Each of these components provides an interface that abstracts it to a higher level. For example, PWM control of the motors in the Motor component provides an interface with the commands setSpeed() and setDirection(). There is no need for the application programmer to follow how the PWM control is implemented, but if a separate motor controller is desired, one can substitute the new Motor component for the old one if the same interface is provided. This flexibility allows the simple substitution of different base platforms as mentioned above in Section 3. The component-based design also provides the most flexibility between the software-oriented roboticist and the application or algorithm-oriented roboticist.

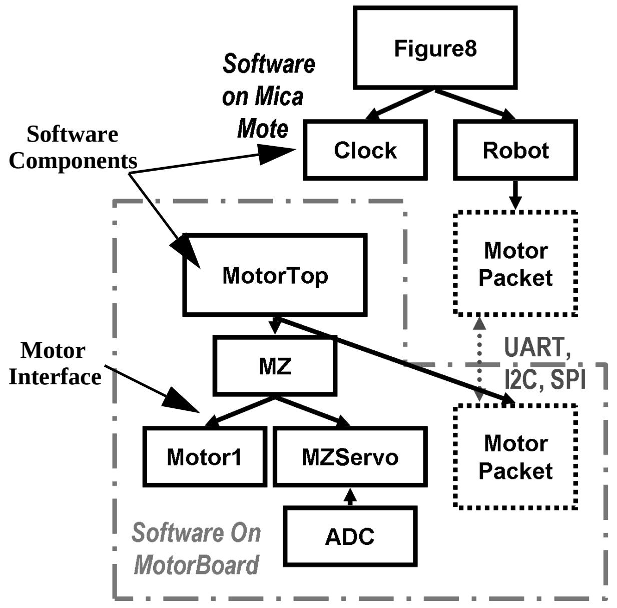

Figure 5. Block diagram of a typical software application. Components (blocks) are plugged together through interfaces (arrows) to build the application.

When using the standard CotsBots base, the Kyosho Mini-Z Racer, it is necessary to implement a simple PI control loop to control the position of the servo to control turning angle. This controller is implemented in the MZServo component and runs at approximately 250Hz.

As mentioned above, all of the low-level drivers are open-source and it is intended that the application programmer program the MotorBoard only once. However, using TinyOS on the MotorBoard allows flexibility in adding new base platforms or new control algorithms for the motors and servos.

4.2 Mica Mote Software

Application software for the CotsBots is written on the Mica mote. In Figure 5, the application graph shows a Figure8 application which uses a Figure8 component, a standard Clock component and the Robot component. In this application, the Figure8 component will receive Clock fire() events and will set the robot speed, direction and turning angle accordingly to drive in a figure8 pattern.

One of the most useful components used on the Mica Mote is the network stack, used for radio communication between robots as well as communication with the MotorBoard. The radio network stack is abstracted using the Active Messaging format described in [8]. A two byte address, one byte group ID, one byte message type, and one byte data length preface a data array for each radio message sent or received.

Communication between boards is handled by the MotorPacket component seen in Figure 5. This component abstracts the actual protocol used so that the user may substitute whichever protocol they prefer. For MotorBoard communication, one byte is used for addressing (MotorBoards may be stacked if more than two motor channels are required), one byte is used for message type and two bytes are given for data.

5. CHARACTERIZATION

Some simple characterization experiments have been used to measure the CotsBots' performance in a typical lab setting. The following tests were performed on a standard tile floor using fresh batteries in the robots.

Speed is set using an 8-bit number (0-255) which controls the pulse-width modulated (PWM) output to the motor. The CotsBots were timed over a distance of

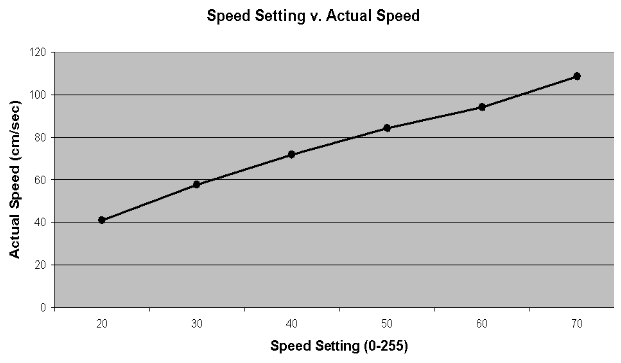

Figure 6. (a) Speed Characterization. Results are fairly linear although speed is heavily affected by battery level.

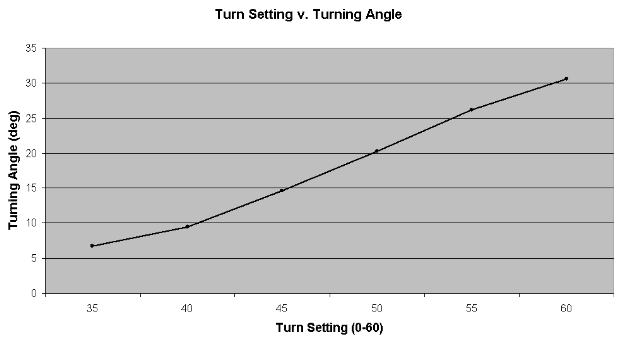

Figure 6. (b) Turning Angle Characterization. Results are symmetric on both sides so only one is shown.

two meters and demonstrated speeds well in excess of 100 cm/s (Figure 6a). Standard deviations were generally around 1-2 cm/s. Unfortunately, the robots are not controllable at speeds much greater than this due to wheel skidding. Speed was relatively linear, but in separate tests it was shown that these speeds vary greatly with battery voltage. However, these deviations could be calibrated out by monitoring battery level with the ATmega8L's ADC on the MotorBoard.

The CotsBots' turning radius was measured by driving the robot in a half circle and measuring the circle diameter. Turns are controlled by a number between 0 and 60, where 30 is considered straight. Results of the equivalent turning angle are graphed above in Figure 6b. Turning angles varied from approximately -30^o to +30^o with standard deviations under 1^o. Results were symmetric for right and left turns and therefore only right turns are graphed in Figure 6b. It is interesting to note that results deviate from linearity when the turning angle is close to straight. In separate experiments it was shown that the turning radius had very little dependence on speed; over a speed setting from 40 to 70, the turning angle changed approximately 1^o.

In addition, several tests were run on the robots using the accelerometer as mentioned above for collision detection. In a typical thirty minute test run, the robot averaged approximately 60 cm/s speed and needed to be reset only 8 times in a typical hallway environment. When the robot needed to be rescued, it was usually due to a very shallow approach into a wall (where deceleration would be minimal) or the robot getting stuck beneath a door. However, these tests demonstrate the viability of an accelerometer for collision detection in the CotsBots.

6. APPLICATIONS

As mentioned earlier, the CotsBots are intended to provide a platform for experimentation and algorithm development in distributed robot systems. To this end, a very simple robot diffusion algorithm is presented. This robot diffusion algorithm is not optimal, but is only a demonstration of what can be accomplished using only very simple sensors and the base functionality of the CotsBots. No localization or distance measurement is required. This algorithm is depicted graphically in Figure 7.

In the BeepDiffusion algorithm, the robots move outward, bouncing off walls until they are out of range of the other robots' beeping. Adjusting the gain on the microphone controls the beep radius and therefore the final distance between robots. The same software is used on each robot and there is no centralized controller involved.

All of the components required for this algorithm, including radio communication, timers, microphone, buzzer, obstacle avoidance, and driving the robot, are already available from [9]. The application writer need only plug these components together to create the application. The BeepDiffusion algorithm has been partially implemented to date and work is currently focused on refining the Microphone component to give more consistent "beep radius" results.

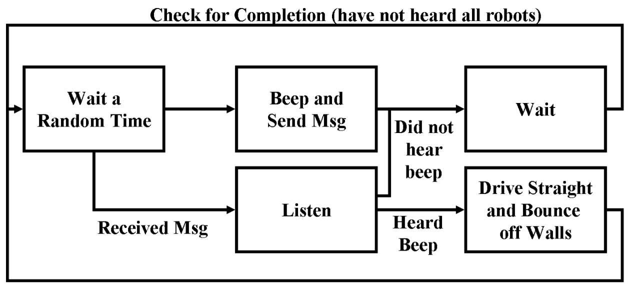

Figure 7. The BeepDiffusion Algorithm. This algorithm is totally distributed and demonstrates cooperation among the robots using only a buzzer and microphone pair.

As sensor platforms for the Mica mote and CotsBots improve, the CotsBots will be able to demonstrate mapping and exploration algorithms as well as group behavior and other algorithms for cooperation. Using robots such as the CotsBots to fill gaps in sensor networks is suggested in [3]. The CotsBots may also be used to mobilize and distribute a sensor network throughout an area or building [14]. Once a standard and easy-to-use platform like the CotsBots has been developed for distributed robotics, it is hoped that algorithms will be more easily ported from simulation to reality.

7. CONCLUSIONS AND FUTURE WORK

An off-the-shelf robot platform for distributed robotics has been presented. The CotsBots are small, inexpensive, off-the-shelf, and provide an extremely flexible hardware and software platform for the user. Hardware is purchased off-the-shelf and only very simple modifications are necessary to build a CotsBots system. In fact, over 80 CotsBots have already been built at a total cost of less than $200 each. Software is open-source allowing the application writer to draw from a large pool of existing components to simplify design. In addition, the software is event-driven which is useful in designing certain robot behaviors. The BeepDiffusion application presented above demonstrates the utility of the base CotsBots platform with only very simple sensors added.

Obviously, much more may be accomplished using the CotsBots. Work has already begun in integrating the robots into a sensor network environment. In this case, the sensor network will provide the localization and sensor feedback information keeping the robot platform as simple as possible. In addition, several additional sensor platforms are currently being investigated for localization and distance measurements. It is hoped that the CotsBots will provide a simple, easy-to-use platform for research in distributed robotics.

8. ACKNOWLEDGEMENTS

We would like to thank Prof. David Culler, his students, and personnel at the Intel Research Laboratory at Berkeley. This research is funded in part by DARPA/SDR and DARPA/NEST.

9. REFERENCES

[1] K-Team, *K-Team:: Khepera II Minirobot.* http://www.k-team.com/robots/khepera/, 2002.

[2] ActivMedia Robotics, *Pioneer 2 General Purpose Robot*, http://www.activrobots.com/ROBOTS/p2dx.html, 2002.

[3] Sibley GT, Rahimi MH, Sukhatme GS. “Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks.” *Proceedings 2002 IEEE International Conference on Robotics and Automation (ICRA).* v.2. pp.1143-8.

[4] McMickell, MB, Goodwine, B, Montestruque, LA. “MICAbot: A Robotic Platform for Large-Scale Distributed Robotics.” *To appear in ICRA 2003.*

[5] Navarro-Serment LE, Grabowski R, Paredis CJJ, Khosla PK. “Millibots.” *IEEE Robotics & Automation Magazine*, v.9, n.4, Dec. 2002, pp.31-40.

[6] NEST group. *NEST Challenge Architecture.* http://today.cs.berkeley.edu/tos/api/arch004.html, 2003.

[7] Crossbow. *Wireless Sensor Networks.* http://www.xbow.com/Products/Wireless_Sensor_Networks.htm, 2003.

[8] Hill, J, et al. “System Architecture Directions for Networked Sensors.” *ACM. Operating Systems Review*. v.34, n.5, Dec. 2000, pp.93-104.

[9] TinyOS Group. *SourceForge.net: Project Info -- TinyOS.* http://sourceforge.net/projects/tinyos/, 2003.

[10] Mainwaring, A., et al. “Wireless Sensor Networks for Habitat Monitoring.” *ACM International Workshop on Wireless Sensor Networks and Applications.* Atlanta, GA, September 28, 2002.

[11] Kyosho, *Kyosho Mini-Z Racers*, http://www.kyosho.com/cars/kyod01x1.html, 2003.

[12] Plantraco, *Plantraco Desktop Rover tracked micro vehicle explorer*, http://www.plantraco.com/product_dtr.html, 2003.

[13] Whitehouse, K. and Culler D. “Calibration as Parameter Estimation in Sensor Networks.” *ACM International Workshop on Wireless Sensor Networks and Applications.* Atlanta, GA, September 28, 2002.

[14] Gage, Douglas W. “Minimum-resource distributed navigation and mapping.” *SPIE Mobile Robots XV.* Boston, MA, November 2000.d