# Pattern Formation Experiments in Mobile Actuator and Sensor Network (MAS-net)

Pengyu Chen, Zhen Song, Zhongmin Wang, and YangQuan Chen
*Center for Self-Organizing and Intelligent Systems (CSOIS),*
*Dept. of Electrical and Computer Engineering, 4160 Old Main Hill,*
*Utah State University (USU), Logan, UT 84322-4160, USA*

***Abstract*— Mobile Actuator and Sensor Network (MAS-net) is a project that adds node mobility and close-loop control concept into the field of Wireless Sensor Network. An experiment platform is built for the MAS-net project. In the experiment platform, cheap, small, and energy-efficient Mica2 motes have been used as both wireless sensors and real-time embedded mobile robot controllers. These mote-based robots are called MAS-motes. An integrated system has been developed to locate MAS-motes by an overhead camera, collect MAS-motes' sensor reading and assign destinations to MAS-motes. This system can communicate with robots via Mica2 motes' built-in radio chips. Pattern formation can bring great benefit to mobile wireless sensor network in sensing range, fault tolerance and sensor-actuator cooperation. This paper tries to use cheap, energy-efficient and mote-based MAS-motes to achieve formation with a given pattern.**

***Index Terms*— Mobile sensor networks, mobile robotics, pattern formation, iterative control, lead-follower formation.**

## I. Introduction

With the growth of internet applications and the advance of mobile computing, wireless networking becomes a very important technology. One special branch in wireless networking research is wireless sensor networks, in which a bandwidth as high as multi-mega bits per second is not required. A node in wireless sensor network usually requires some sort of sensing to the environment and coordinating with each other. A very thorough overview of the wireless sensor networks can be found in [1]. The Mobile Actuator-Sensor Network (MAS-net) project combines mobile robotics with the wireless sensor networks. The objective is to develop systems that can collect information and respond to the spatially distributed diffusion processes. An extended application of this project can be in homeland security, where chemical, biological, radiological or nuclear (CBRN) terrorism can cause devastating damages. It is thus important to have a system that can respond and control the diffusion process of the harmful materials. Some research challenges and opportunities are presented in [2], [3], [4]. A preliminary result of a diffusion-based path planning is given in [3].

The rest of this paper is organized as follows. Section II describes the two communication modules used in the system. Section III describes the function of the image processing module that identifies each robot and gives the mobile robot's position and orientation. Section IV describes the implementation of pattern formation from the underlying movement control. Some experiment results are given in Section V. Finally, section VI concludes the paper.

Corresponding author: Prof. YangQuan Chen, Center for Self-Organizing and Intelligent Systems, Dept. of Electrical and Computer Engineering, 4160 Old Main Hill, Utah State University, Logan, UT 84322-4160. **T**: (435)7970148, **F**: (435)7973054, **W**: `www.csois.usu.edu`, **E**: `yangquan.chen@usu.edu`

### A. MAS-net test-bed

A test-bed has been built to observe the 2D diffusion process. The platform is a $92.7 \times 141.25$ square inches container, covered with transparent acrylic boards. Ten small Mica2-based robots, as shown in Fig. 1, are built to move on top of the platform [5], [6]. These robots are called MAS-motes. The application scenario is to observe, detect and control the diffusion boundary when a 2D diffusion process, such as fog, is released into the platform. An overhead camera is hung about 72 inches above the platform. The function of the camera is to identify each MAS-mote and to give the position and orientation of the MAS-mote. The image processing is performed in a central computer (PC Host or base station) by a subsystem named "pseudo-GPS". The pseudo-GPS, message transmission and reception, data collection, decision making, and graphic user interface are all integrated into one program called "Integrated Control System(ICS)". The platform is illustrated in Fig. 2. The ICS is running on the PC Host.

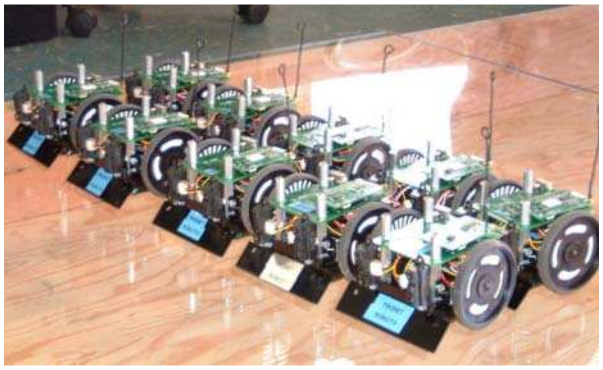


Fig. 1. Ten Mica2-based robots for the MAS-net project.

### B. MAS-motes

The MAS-motes are based on Mica2 motes developed by CrossBow [6]. It has an ATmega 128L as its central processor. A low power FSK (Frequency Shift Keying) RF transceiver chip, CC1000, is also integrated in the Mica2 board [7]. An embedded operation system, TinyOS, developed by UC

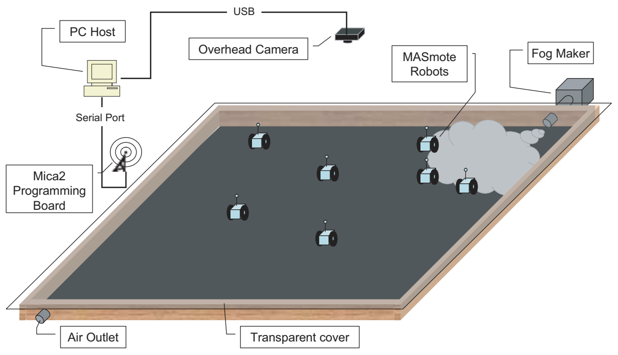


Fig. 2. Illustration of the test-bed for the MAS-net project.

Berkely is specially designed for Mica2 motes and their family [8]. TinyOS is an event-driven operating system developed in nesC, a C-based programming language [9] [10]. TinyOS only supports non-preemptive multi-tasking. It sacrifices complex preemptive scheduling and inter-process communication for simplicity and tiny code size. However, non-preemptive scheduling hinders TinyOS from hard real-time applications.

MAS-motes are two-wheel differentially steered robots. With the dimension only 9.5 cm $\times$ 9.5 cm $\times$ 6.5 cm, they have plenty of space to move around on our platform. The robots are all equipped with a photo resistor facing down to detect the concentration of fog where the robot stands. MAS-motes are driven by two servomotors. Each servomotor has a built-in servo controller. The wheels attached to them are 6.6 cm in diameter. To reduce the cost of encoders, we have made our own encoders to support the odometer of MAS-motes. The encoder patterns are drawn by Matlab and then printed on normal papers. They are then glued onto the inner side of wheels. The encoder pattern only has 32 segments, which gives MAS-motes a resolution about 11 degrees per strip and about 0.6 cm in length. Two photo-reflexive sensors are mounted on the MAS-mote's body to detect the white and black band on the encoder pattern. The two photo-reflexive sensors are connected to ADC channels of ATmega 128L.

## II. Communication

All MAS-motes communicate via wireless communication. A mote mounted on programming board works as a gateway between wireless communication and serial port communication to PC. Its only purpose is to forward all messages between the serial port and the RF port.

Communicating with the gateway mote is an integrated function of the ICS. The protocol for serial communication is compliant with the original design of TinyOS. The protocol supports framed packets, flow control, and CRC check. In ICS, a transceiver thread is dedicated for enforcing the protocol and communicating with the gateway mote. The transceiver thread is designed so that no polling or busy waiting is needed in the system and the system is signaled immediately when an event occurs.

Controlling MAS-motes may need very frequent messages from the pseudo-GPS with very little latency tolerance. On the other hand, control commands need more reliable connection but can tolerant some degree of latency. Currently, S-MAC is used to replace the Berkeley MAC layer in TinyOS [11]. When unicasting, the S-MAC can support more reliable communication by performing RTS-CTS and ACK-Retransmission [12], [13]. While the reliability is improved, the control overhead can be very high. Therefore, commands, which need a more reliable connection, are unicasted to the desired MAS-motes. Pseudo-GPS messages, which include position information of MAS-motes and have less tolerance to latency, are broadcasted to all MAS-motes.

## III. Pseudo-GPS

The function of the pseudo-GPS module is to provide the position, orientation, and ID of each MAS-mote based on the markers on top of MAS-motes. The procedure of the pseudo-GPS includes color segmentation, marker identification and coordinate transformation. The color segmentation is based on HSB color model so it is less affected by illumination. ARToolKit is modified and adopted into our system for marker identification [14]. A method from [15] is used to transform the positions of MAS-motes from image coordinates to world coordinates. The pseudo-GPS can give positions and orientations with an acceptable precision. The maximum error is less than 3mm in position and 3 degrees in orientation. The pseudo-GPS can operate as fast as 6 frames per second in our workstation, which has two Intel Xeon HT processors and 1GB RAM. Considering the network traffic, currently the pseudo-GPS messages are broadcasted every 500ms. Up to four pseudo-GPS messages can be packed together into one packet. The network traffic can be greatly reduced in this way.

## IV. Pattern Formation

The pattern formation is implemented by the leader-follower behavior. The followers periodically update the leader's position and calculate their own desired position according to pseudo-GPS messages. To show how our pattern formation is implemented, it is necessary to first describe how the MAS-motes move to their destinations.

### A. Odometer

As described in Section I-B, the encoders on MAS-motes are made by ourselves. The two photo-reflexive sensors for detecting the white and black bands on the encoder pattern are connected to ADC channels of the ATmega 128L microcontroller. In ATmega 128L, the multiple ADC channels can only work in serial sequence, which means only one ADC channel is working at a time. Another ADC channel is triggered after the data conversion of the previous ADC channel has finished. In TinyOS, ADC channels are working in asynchronous fashion. After a task triggers the ADC channel, it will have to move on to other work. When the data is ready, TinyOS will launch another task to handle it. We can only process the photo sensor and trigger the next ADC channel in the handling task. Since TinyOS is non-preemptive, when the handling task can actually be active is unpredictable. Therefore, it is possible to skip some strips. More importantly, it is difficult to obtain accurate linear and angular velocity of MAS-motes. Currently, the odometer of MAS-motes can only provide position feedback. The kinematics of our MAS-motes is:

$$\begin{aligned} x(k+1) &= x(k) + \frac{Cr_l + Cr_r}{2}\cos(\theta(k)) \\ y(k+1) &= y(k) + \frac{Cr_l + Cr_r}{2}\sin(\theta(k)) \\ \theta(k+1) &= \theta(k) + \frac{Cr_r - Cr_l}{d} \end{aligned} \quad , \qquad (1)$$

where $r_l$ and $r_r$ are the passed strips the sensors has detected and $C$ is a constant to convert number of strips to the distance that wheel has moved. Since the two wheels have identical dimensions and encoder patterns, they should have the same conversion constant. Equation (1) will have larger error if the interval of the calculation is not short enough. Errors can come from inaccurate $C$, $r_l$, and $r_r$ and errors will accumulate so the position information from pseudo-GPS is very important.

## *B. Control Stages*

As a summary of how MAS-motes are controlled, a flow chart of the control program is shown in Fig. 3. Every destination command received by a MAS-mote will be put into a task queue. The MAS-mote gets new destination from the task queue if it has no destination now or it has reached the current destination. If there is no task in the task queue, the current destination will remain unchanged. Some emergency commands are also implemented. An AC_STOP command stops a MAS-mote and clears the MAS-mote's task queue and current destination. The AC_PAUSE just stops the MAS-mote until an AC_RESUME is received. In Fig. 3, the $e_a$ is the angular error between a MAS-mote's orientation and the desired orientation; $e_p$ is the position error, i.e. the distance from the MAS-mote to its destination. There are three stages in a MAS-mote's movement to accomplish a task. They are listed below sequentially.

- INIT_TURN: This is the initial spin-in-place before moving toward the destination. At this stage, $e_a$ is the angular error between the MAS-mote's orientation and the direction towards the MAS-mote's destination. The purpose of this stage is to minimize $e_a$ before the MAS-mote moves towards the destination.
- RUNNING: This is the Point-to-Point control, during which MAS-motes are moving toward their destinations. A PI controller is used for this stage.
- END_TURN: This is the spin-in-place when a MAS-motes is at the destination. At this stage, $e_a$ is the angular error between a MAS-mote's orientation and the desired orientation assigned to this MAS-mote by the ICS's destination command. If there is no task in the task queue and the last task has assigned a desired orientation for the MAS-mote, the MAS-mote needs to turn to the assigned orientation at this stage.

The control loop is triggered every 15ms. Every time the control loop starts, the MAS-mote updates its odometer and recalculates $e_a$ and $e_p$. Then, the MAS-mote decides whether it is necessary to transit stage. The transition of stages depends on $e_a$ and $e_p$ as shown in Fig. 3. Normally, a MAS-mote goes through the three stages from INIT_TURN, RUNNING, and finally to END_TURN to accomplish a task. However, there are some special cases. When $e_a$ is too large at the RUNNING stage, the MAS-mote stops and transits back to the INIT_TURN stage. Therefore, the MAS-mote will perform a

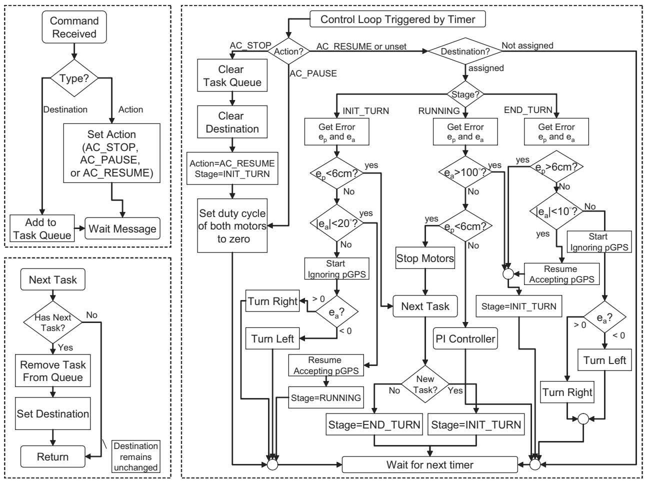


Fig. 3. Flow chart of MAS-mote control.

spin-in-place to correct its orientation if its the orientation has deviated too large. This behavior can reduce the settle time of a MAS-mote. At the END_TURN stage, $e_p$ is always checked to ensure that the MAS-mote is at the correct position in case the position is changed by the spin-in-place behavior.

## *C. Iterative Control*

A MAS-mote may stop at an incorrect position because the inaccurate odometer may indicate that the MAS-mote has reached the destination. When a MAS-mote is moving, the pseudo-GPS information can have larger error because of the system latency. Our system amends this error by the feature of iterative control. As described in Section IV-B, the current destination of a MAS-mote does not change unless there is an unexecuted task in the task queue or an AC_STOP is received to reset the destination. Furthermore, the control loop always checks $e_a$ and $e_p$ and adjusts the MAS-mote's position and orientation when necessary. Therefore, a pseudo-GPS message may trigger the received MAS-mote to adjust its position and orientation even after the odometer of the MAS-mote indicates that the destination has been reached. Since the pseudo-GPS messages are sent periodically, MAS-motes will have the effect of iterative control. The observation of the pseudo-GPS is more accurate then that of the built-in odometer. Besides, when a MAS-mote is stopped, its position and orientation in the pseudo-GPS message is even more accurate. Therefore, the iterative control triggered by pseudo-GPS can drive a MAS-mote very close to its destination. The feature of iterative control is also helpful when a MAS-mote is moved by an undesired outside force, such as collision by another MAS-mote. The MAS-mote can automatically go back to its desired position and orientation.

## *D. Point-to-Point Control*

Based on the position feedback, a PI controller is used for the MAS-mote's Point-to-Point control in the RUNNING stage. Our strategy is to maintain the MAS-mote's heading toward its destination and let the MAS-mote move forward until the MAS-mote reaches the destination. When $e_a$ is zero, the PWM signals for the two motors are both 50 percent duty cycle. By convention, the angle increases in counter clockwise. When $e_a$ is positive, the left motor is assigned a higher duty cycle while the right motor is assigned a lower duty cycle. The MAS-mote will deviate to the right. Similarly, when $e_a$ is negative, the duty cycle is assigned so that the MAS-mote deviates to the left. When $e_a$ is too big, the controller output may saturate the motors and cause the integrator windup phenomenon, which causes a large overshoot and slow settling time. Our control algorithm is designed with an anti-windup approach [16]. The control simulation model is shown in 4. The control parameters KP and KI are designed via simulation

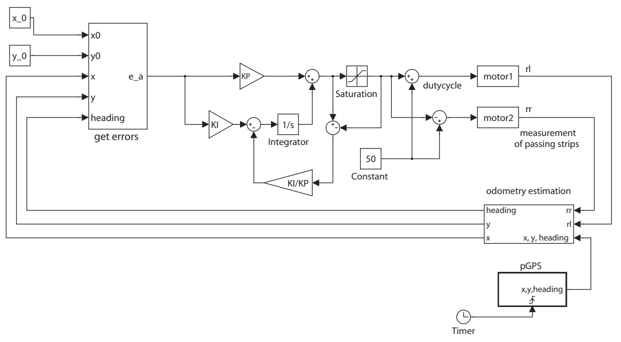


Fig. 4. PI controller for point-to-point control.

and then fine tuned by experiments. However, optimal KP and KI are different for each MAS-mote. They are also subject to voltage change of the batteries. When the motors are working, the inherent friction is also changing. Therefore, it is very difficult to find a set of good KP and KI for each MAS-mote. This problem and the inaccurate encoder make our MAS-motes usually unable to go an ideally straight line. Therefore, the help from pseudo-GPS information is very important.

### E. Spin in Place

At the stages of INIT_TURN and END_TURN, spin-in-place behavior turns a MAS-mote to the desired orientation. The spin-in-place is implemented by setting a constant duty cycle to the two motors but with different directions according to the sign of $e_a$. As shown in Fig. 3, spin-in-place only relies on the MAS-mote's built-in odometer for feedback. The MAS-motes turns until the odometer shows that it has achieved the desired orientation. During this period, pseudo-GPS messages are ignored. This is because the spin-in-place behavior starts and finishes in a very short period. The position and orientation observed by the pseudo-GPS have delay error and therefore is not necessarily more accurate than the odometer in the short period. In our experiments for spin-in-place behavior, significant overshoots are observed if the MAS-motes are accepting pseudo-GPS information. The INIT_TURN stage does not require a good precision because its purpose is only to reduce $e_a$ before moving. The PI controller in Point-to-Point control will try to compensate that error. But a good precision is more desirable for the END_TURN stage since the orientation is commanded by the ICS. When a spin-in-place is finished, the MAS-mote resumes accepting pseudo-GPS messages. The iterative control effect of our design will become active. The MAS-mote will correct the orientation when it receives the next pseudo-GPS message.

### F. Leader-Follower Behavior

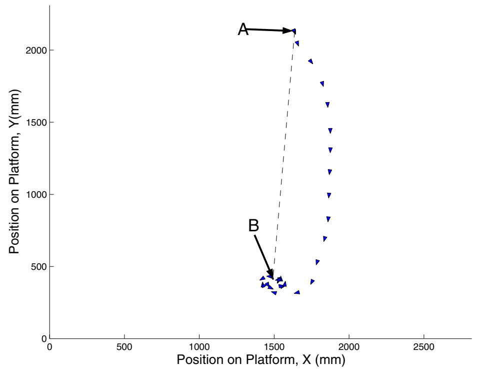
Our approach for formation control exploits the periodically broadcasted pseudo-GPS messages. The formation control is actually a leader-follower behavior. The Integrated Control System(ICS) can assign any MAS-mote(s) to follow any other specified MAS-mote. Followers can have any relative position to their leader. The parameters for followers are $(r_f, \theta_f)$, where $r_f$ is the distance to the leader and $\theta_f$ is the angle from the leader's heading. Followers get the $(r_f, \theta_f)$ from the Follow command issued by the Integrated Control System. Currently, the desired orientations of followers are the orientation of their leader. The Followers get the leader's position and orientation from broadcasted pseudo-GPS messages. To maintain the formation, followers calculate their own desired states $(x, y, \theta)$ from the leader's pseudo-GPS messages by equation (2).

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} r_f \cos(\theta_f + \theta_l) \\ r_f \sin(\theta_f + \theta_l) \\ 0 \end{bmatrix} + \begin{bmatrix} x_l \\ y_l \\ \theta_l \end{bmatrix}, \qquad (2)$$

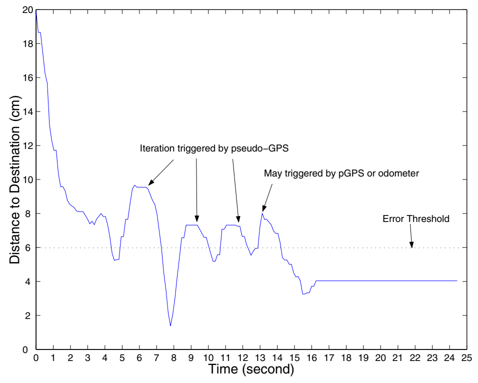
where $(x_l, y_l, \theta_l)$ are the state of the leader. Equation (2) is performed every time the follower receives the pseudo-GPS message of the leader. The desired positions for followers can change dramatically, especially when the leader is performing the spin-in-place behavior. The pseudo-GPS messages are broadcasted, which are more likely to be lost because the broadcasting does not practice RTS-CTS and ACK-Retransmission mechanism of the S-MAC. Even if they are received by the followers successfully, the delivery time is not guaranteed. For all these reasons, the followers usually lag behind the movement of their leaders. Currently, when the followers are too far away (15 cm) from their desired positions, followers will send a WAIT_4_ME message to their leaders. The message has the same effect to the leader as AC_PAUSE when the leader will stop moving. When followers reach the desired positions, they send WAIT_4_ME message again with a cancel flag set to cancel its wait request. The leader keeps a counter for how many followers it should wait. When the counter is zero, it will resume its movement. In the leader-follower mode, the iterative control is still active. Followers always check and adjust their position and/or orientation when necessary.

## V. Experiments

An experiment result is shown in Fig. 5. In this experiment, a MAS-mote is commanded to go from point A to point B. Figure 5(a) is the trace of the MAS-mote's movement. Obviously, this MAS-mote cannot go a straight line. However, with the help from the pseudo-GPS and the iterative control, the MAS-mote can still reach the destination. Figure 5(b) records the change of the MAS-mote's $e_p$ with time. The error threshold for $e_p$ is set at 6 cm, which means when $e_p$ is less than 6 cm, the MAS-mote is considered at the destination. Only the last 20cm toward destination is shown in Fig. 5(b) so the behavior at the end is more clear. Some long delays are observed around the 6-th, the 9-th, and the 11-th second in Fig. 5(b). They are evidences of pseudo-GPS triggered corrections. If the corrections are triggered by the MAS-mote's built-in odometer, there should be no long delays. Another experiment for spin-in-place is shown in Fig. 6. A MAS-mote is commanded to stay at its original position but turn its orientation from about -178 degrees to 0 degree. Figure 6 shows the MAS-mote has 20 degrees of overshoot after its first turn, during which the MAS-mote is ignoring pseudo-GPS. There are three obvious steps after the first turn. These three steps mean some amount of delay before correction so they represent three iterations triggered by pseudo-GPS messages. The iterative control is effective in this experiment, too. An experiment for pattern formation is shown in Fig. 7, where a leader is placed at (755 mm, 1420 mm) with 43.5 degrees of orientation. Other MAS-motes are initially placed near the left edge of the platform. From top to bottom, they are labelled as "Follower 1" to "Follower3" commanded to follow the leader with parameters $(r_f, \theta_f)$ equal to (400 mm, 0 degree), (400 mm, 120 degree) and (400 mm, 240 degree), respectively. Figure 7(a) is the trace record of the three followers forming the pattern formation. After the formation is formed, the leader is commanded to move toward the right side of the picture and then move toward the bottom of the picture a little to show how the formation turn. The position errors of the three followers in Fig. 7(b) are shown in Fig. 8. The leader's movement is shown by its position difference between two pseudo-GPS samples. The plot of the leader only intends to show the time during which the leader is moving. The leader receives the command and starts moving at the time point A in Fig. 8. The leader waits the followers from time points B to C. The leader waits again between time points D and E. At the time point F, followers ask the leader to wait again. The leader stops but it happens to be at the destination of the leader so the leader never moves again. Followers eventually catch up with the leader and regain the formation. In Fig. 7(c), the formation can still be obtained after the leader turned its direction to the bottom of the picture and moved a little.



(a) Movement trace in an iterative control.



(b) Position error in the movement

Fig. 5. One MAS-mote movement

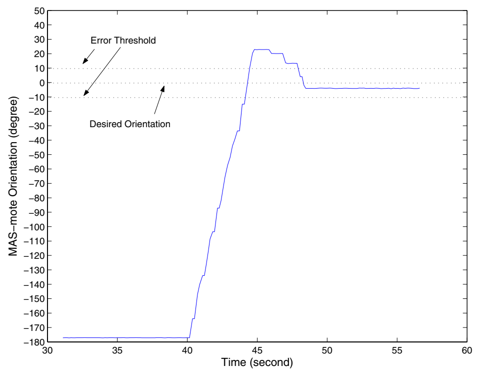


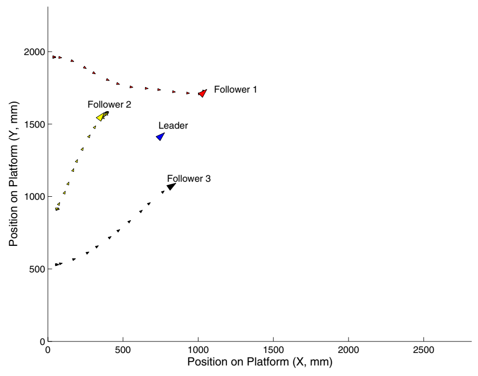
Fig. 6. Spin-in-place under iterative control.

## VI. Concluding Remarks

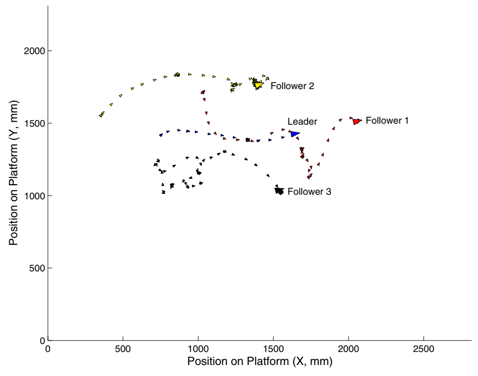
In this paper, the MAS-motes are able to maintain a formation by observing the position information and coordinating each other. Although the formation can not be maintained very well during movement, the formation can always be obtained after the leader arrives its destination. Therefore, there will be a MAS-mote formation at the desired location with desired orientation and pattern. This research also tries to push the limit of current works of wireless sensor networks by doing robot formation on the most widely used platform, Mica2 motes. The most significant bottleneck we experienced in this research lies in wireless communication. The need for low-power consumption motes has limited the ability for coordinating robots. Through this research we have laid a foundation, gained experience and created many new ideas for future researches of the MAS-Net project.

### A. Future Work

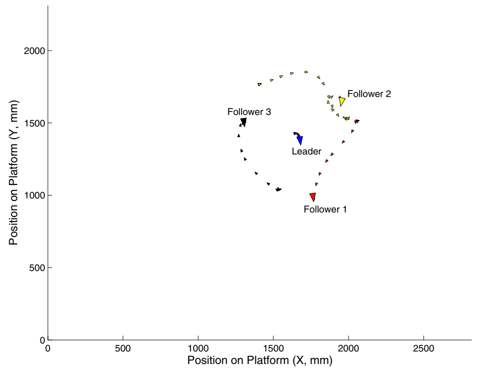
Real-time operating system and communication protocol should be used to have deterministic latency in the system. The accuracy of the pseudo-GPS and encoder and responsiveness of the controller can be improved in real-time system. In the future, the MicaZ with ZigBee wireless communication protocol will be adopted so that the wireless communication can have wider bandwidth, less and deterministic latency and more reliable connection [17], [18]. Kalman filter can be used to combine the observation from pseudo-GPS and the built-in odometer [19] and the experimental result can be even more accurate and smooth. Some researches have been done in the localization by RF signal strength [20], [21], [22], [23]. This approach can be adopted in our system to replace pseudo-GPS. All decision making can be distributed to MAS-motes and eventually eliminate the role of the central computer.



(a) Pattern formation trace.



(b) Formation moves to right.



(c) Formation turns.

Fig. 7. Formation movement trace.

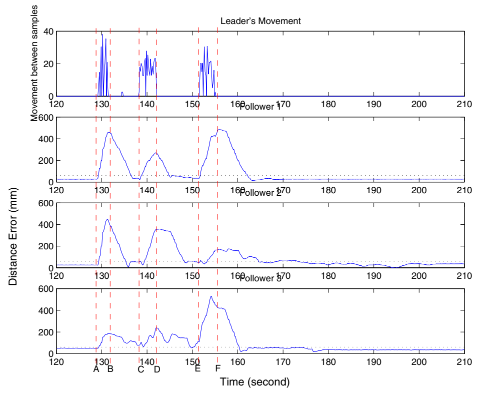


Fig. 8. Follower's position error during the movement of Fig. 7(b).

All MAS-motes will serve a common mission collectively by coordinating each others independently.

## References

[1] E. H. Callaway, *Wireless Sensor Networks: Architectures and Protocols*. 2000 N.W. Corporate Blvd., Boca Raton, Florida 33431: CRC Press LLC, 2003.

[2] Y. Q. Chen, K. L. Moore, and Z. Song, "Diffusion boundary determination and zone control via mobile actuator-sensor networks (MAS-net) – challenges and opportunities," in *Proc. of SPIE Conf. on Intelligent Computing: Theory and Applications II, part of SPIE's Defense and Security*. Orlando, FL., USA: SPIE, April 2004.

[3] K. L. Moore, Y. Q. Chen, and Z. Song, "Diffusion based path planning in mobile actuator-sensor networks (MAS-net) - some preliminary results," in *Proc. of SPIE Conf. on Intelligent Computing: Theory and Applications II, part of SPIE's Defense and Security*. Orlando, FL., USA: SPIE, April 2004.

[4] K. Moore and Y. Chen, "Model-based approach to characterization of diffusion processes via distributed control of actuated sensor networks," in *Proc. of the IFAC Symposium of Telematics Applications in Automation and Robotics, Helsinki University of Technology Espoo, Finland*, 2004.

[5] Z. Wang, Z. Song, P. Chen, A. Arora, D. Stormont, and Y. Q. Chen, "MASmote – a mobility node for MAS-net (Mobile Actuator Sensor Networks)," in *Proc. of the First IEEE Int. Conf. on Robotics and Biomimetics, Shengyang, China*, 2004.

[6] Crossbow Technology Inc., "Mica2 Series," `http://www.xbow.com`, 2004.

[7] Chipcon Inc., "CC1000 Product Information," `http://www.chipcon.com/`.

[8] Wireless Embedded Systems Lab, UC Berkeley, "TinyOS," `http://www.tinyos.net/`.

[9] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation, San Diego, California, USA*, 2003, pp. 1–11.

[10] David Gay and Philip Levis and David Culler and Eric Brewer, "nesC Language Reference Manual," `http://www.tinyos.net/tinyos-1.x/doc/nesc/ref.pdf`.

[11] ISI Laboratory for Embedded Networked Sensor Experimentation, "MACSS: MAC Protocols Specific for Sensor Networks," `http://www.isi.edu/ilense/macss/index.html`.

[12] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[13] ——, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, NY, USA*, 2002.

[14] Human Interface Technology Laboratory of The University of Washington, "ARToolKit," `http://www.hitl.washington.edu/artoolkit/`, 2004.

[15] C.-C. Wong, C.-P. Huang, B.-C. Lin, and Y.-S. Huang, "Position correction in the image process of robot soccer game," in *Proc. of The Seventh Conference on Artificial Intelligence and Applications (TAAI2002), Taichung, Taiwan*, 2002, pp. 684–689.

[16] H.-B. Shin, "New antiwindup PI controller for variable-speed motor drivers," *IEEE Trans. of Industrial Electronics*, vol. 45, no. 3, pp. 445–450, Jun 1998.

[17] ZigBee Alliance, " `http://www.zigbee.org/`," .

[18] G. Legg, "ZigBee: Wireless technology for low-power sensor networks," [`http://www.techonline.com/community/tech_topic/36561`].

[19] R. Negenborn, "Robot localization and kalman filters," Master's thesis, Utrecht University, 2003.

[20] P. Bahl and V. N. Padmanabhan, "Radar: An in-building RF-based user location and tracking system," in *Proceedings of the IEEE Infocom 2000, Tel-Aviv, Israel*, vol. 2, Mar 2000, pp. 775–784.

[21] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal, "Ad-hoc localization using ranging and sectoring," in *Proceedings of IEEE INFOCOM '04, Hong Kong, China*, April 2004.

[22] K. Whitehouse, "Calamari: a sensor field localization system," [`http://www.cs.berkeley.edu/~kamin/calamari/`].

[23] K. Lorincz and M. Welsh, "A robust, decentralized approach to RF-based location tracking," Harvard University, Tech. Rep. TR-19-04, 2004.