

Using Real-Time Logic Synthesis Tool to Achieve Process Control over Wireless Sensor Networks

Jianping Song, Aloysius K. Mok
Dept. of Computer Sciences
University of Texas at Austin
Austin, TX 78712
{sjp, mok}@cs.utexas.edu

Deji Chen, Mark Nixon
Emerson Process Management
12301 Research Blvd, Building III
Austin, TX 78759
{deji.chen, mark.nixon}@emersonprocess.com

Abstract

Wireless sensor networks have been a very active research field in the past few years. However, most of extant research has focused on wireless sensing, such as environmental and habitat monitoring [6]. In this paper, we investigate the feasibility of applying wireless technologies in industrial real-time process control systems. We define a minimum set of assumptions about the underlying sensor networks in order to achieve real-time support for process control. These assumptions are necessary and practical from an industrial perspective. We formulate the modeling of a wireless process control configuration as a multi-processor scheduling problem. The resulting scheduling problem is solved by the scheduler synthesis tool MSP.RTL [7] to produce a valid configuration for deployment. The deployment, because of the way it is derived, will provide real-time wireless support for the controlled processes. Simulation results on data from real plant configurations show that this approach can also drastically reduce potential interferences between wireless transfers.

1. Introduction

Wireless sensor networks have been an active research topic during the past few years. Researchers have conducted extensive studies on various aspects of wireless sensor networks: information dissemination [4], energy-efficient routing [1] and security [8], et al. In addition, wireless sensor networks have been successfully applied to environmental and wildlife habitat monitoring [6].

As its name implies, wireless sensor networks are mostly used for “sensing”, that is, collecting data and feeding the data to a central processing point. Several industrial organizations, such as WINA [11] and ZigBee [12], have been advocating the application of wireless technologies in industrial control. For example, ZigBee Specification V1.0 was ratified in late 2004 and ZigBee compliant products are readily available on the market [13]. Industrial process control has real-time requirements

for deterministic data transfers. Current sensor networks by their nature have failed to meet this need.

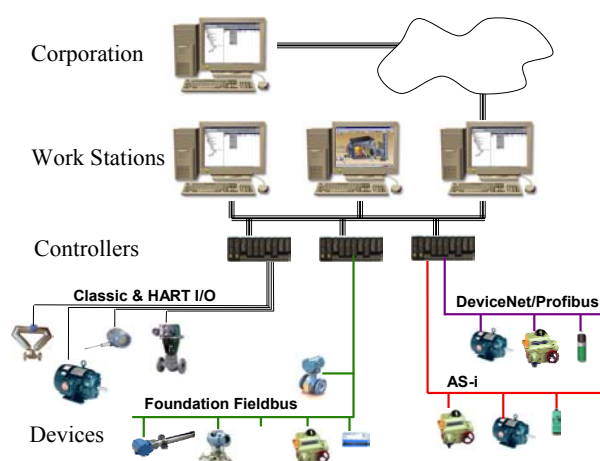


Figure 1: A traditional process control system

A traditional process control system is shown in Figure 1 [2]. As illustrated in this figure, devices are connected to controllers via some types of buses, such as Fieldbus [9] and Profibus [10]. Usually, each device is either a sensor or an actuator. A sensor collects certain status information of a process and feeds it to a controller. Based on the readings from sensors, the controller determines whether the actuators should act to maintain some physical property of the process, e.g., the flow in a pipe must be kept at a fixed speed. In this case, the tasks on sensors, controllers and actuators comprise a control loop. The control loop is executed periodically to maintain the constant flow in the pipe. There can be multiple control tasks running on a controller. Users can interact with controllers via workstations which are connected to the corporate network.

In a wireless process control system, the buses between controllers and devices are replaced by wireless networks. Although the benefit of wireless process control is very attractive, many technical issues have to be resolved to make wireless process control feasible:

- **Transient Interferences:** wireless communication is sensitive to interferences. We need to provide deterministic and timely data delivery in case of temporary wireless link failures.
- **Power Efficiency:** Energy is always a problem for battery-powered devices. The control system should not fail if one sensor or actuator is depleted of energy.
- **Security:** Some sensor readings are sensitive and need to be protected from eavesdropping. Also, the actuator should be shielded from intruders to avoid unintended actions.

This paper tries to address some of the challenges in wireless process control by first making some reasonable assumptions about the wireless network such as synchronized time among the sensors and an upper limit on transmission retries. We shall schedule all the tasks and communications in a wireless process control system through a multi-processor schedule synthesis tool, MSP.RTL [7]. On the one hand, the schedule created by MSP.RTL guarantees real-time performance of the control system. On the other hand, if two communications are known to interfere with each other (such as sharing a common endpoint), a constraint is created for MSP.RTL to ensure the communications run in a mutually exclusive mode. As a result, potential interferences between wireless communications can be reduced dramatically.

Our contributions in this paper are as follows:

Exploration of real-time control over wireless networks: In this paper, we focus on implementing real-time control, instead of sensing, on wireless networks.

Translation of a wireless process control configuration to a multi-processor scheduling problem: We model wireless process control configuration as a scheduling problem, which can be solved by existing scheduling tools.

Consideration of wireless interferences in the translation procedure: Wireless interferences are tolerable in the resulting configuration deployment by our design.

Evaluation on real plant configurations: We evaluate our approach on a wireless process control system derived from a real plant configuration based on the Foundation Fieldbus. As Fieldbus is widely used in current process control systems, our contribution is to find a feasible path to transition from wireline process control systems to wireless process control systems which provides real-time support and fault tolerance capabilities.

The rest of the paper is organized as follows. Section 2 introduces the MSP.RTL tool. In Section 3, we formulate process control loop configurations used in this paper. Section 4 presents the characteristics of the ZigBee-based wireless sensor networks. Based on the problem formulation in these three sections, we describe our

deployment methodology by applying the scheduling tool MSP.RTL to wireless process control in Section 5. Section 6 shows simulation results on data from real plant configurations. We conclude this paper in section 7.

2. The MSP.RTL tool

MSP.RTL is a tool for producing cyclic schedules for multiprocessor systems with a wide variety of timing constraints [7]. The input to MSP.RTL can be customized for different application domains, as long as their timing semantics can be expressed in RTL (real time logic). A typical input to MSP.RTL contains three sections: processor section, process section and constraint section. All processors in the system are listed in processor section. The process section presents details about each process that include its identification, release time, execution time and period. All timing and resource requirements are defined in the constraint section. In one single constraint, there can be multiple options, of which only one can be selected. List 1 shows a simple input file to MSP.RTL. In the sample input file, there are one processor (P1) and four processes (A, B, C, D). Based on the constraints in the input file, the execution of processes A, B and C are mutually exclusive.

The MSP.RTL tool searches for a feasible schedule in a three step process: the first step constructs a temporal constraint graph representing the input timing specification. The second step finds a set of solutions of the temporal constraint graph without considering the resource constraints. The third step searches through the set of solutions to derive a feasible schedule which satisfies the resource constraints. The output for the sample input in List 1 is shown in List 2.

3. Process control configuration

A basic process control unit consists of some inputs, a control algorithm, and some outputs. Periodically, the algorithm reads the inputs, performs some calculation, writes to the outputs, and sometimes receives feedback from the outputs. These steps constitute a control loop. Figure 2 is an example of a PID control loop. In this figure, AI1 is an input function block, which samples the process and produces input data. PID1 is a proportional-integral-derivative function block. When executed, PID1 produces output values based on multiple parameters including the input value. AO1 is an output function block which actuates the device to the designated output value. The loop must be executed periodically to maintain the process in a steady state. Fast processes such as pipe flows require fast loop execution rates; slow processes such as tank storages can tolerate low loop execution frequencies.

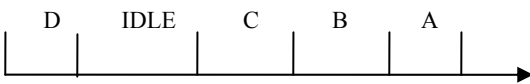
PROBLEM exp1
 PROCESSOR P1

PROCESSES
 A (P1) 20 40 110
 B (P1) 20 60 90
 C (P1) 20 50 91
 D (P1) 20 0 120

CONSTRAINTS
 { [@(_A) <= @(^B)] OR [@(_B) <= @(^A)] }
 AND { [@(_A) <= @(^C)] OR [@(_C) <= @(^A)] }
 AND { [@(_B) <= @(^C)] OR [@(_C) <= @(^B)] }

List 1. A, B, C are mutually exclusive on P1

t1=0 t2=20 process[D]
 t1=20 t2=50 process[IDLE]
 t1=50 t2=70 process[C]
 t1=70 t2=90 process[B]
 t1=90 t2=110 process[A]



List 2. Final schedule for A, B, C and D

In some occasions, there may be data links between two loops, but a loop is usually scheduled independently at its own period. Experienced configuration engineers are needed to produce good loop configurations from a plant process requirement.

4. ZigBee-based wireless sensor networks

Of all the industrial specifications, ZigBee [12] seems to be the most promising for process control. ZigBee is targeted at low-power and low data rate (250kb/s) communication environments. There are three types of devices in a ZigBee network: controllers, routers and end devices. A controller can be a router, whereas an end device can not be a router. The maximum communication range for a ZigBee network can be as long as 100 meters. Currently, ZigBee devices can have about 100 days of battery life.

Although the features listed above make ZigBee an attractive option for wireless process control, it does not entirely fit process control as one would hope. For example, control in ZigBee networks is mainly event triggered so that sensors sleep most of the time when there are no events, whereas a process control loop must be executed periodically.

In this paper, we shall focus on wireless process control systems built on top of ZigBee-like networks. In addition, we make some assumptions about the underlying networks in order to achieve real-time control:

1. The clocks of all devices in a network are synchronized. Several synchronization protocols for sensor networks [3, 5] can be readily applied to process control systems.
2. There is a way to program the sensors/actuators, either through some programming ports or wireless links. As the schedule is generated offline by MSP.RTL, the devices have to be programmed with the schedule.

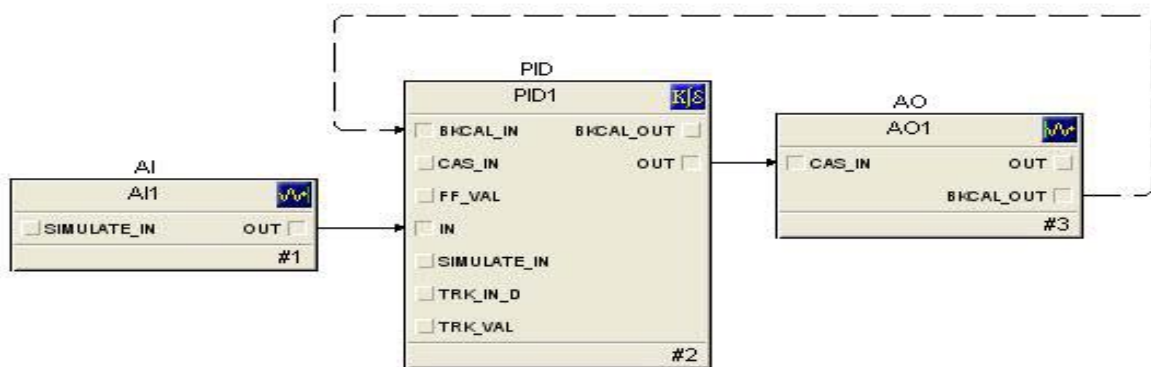


Figure 2. A PID control loop

- There is an upper limit on failed retries, i.e., the data transmission is assumed to eventually succeed after limited retries. Without a limit we can never achieve real-time control, even for hardwired systems. If the limit is exceeded, recovery action must be taken. We shall not address this type of recovery in this paper.

Assumption 1 is essential for the network to provide real-time services and is widely enforced in wireless sensor networks, as people always want to get a meaningful timestamp on each triggered event. The development kit coming with ZigBee devices provides the capability stated in assumption 2. Assumption 3 is also reasonable because modern wireless technologies such as spread spectrum and adaptive radio could handle interferences fairly efficiently.

Based on assumption 3, only transient interferences are considered in this paper. We cope with transient link failures in two ways. First, two communications which share an end node will be scheduled in non-overlapping time periods. This arrangement would avoid most hidden terminal problems, a major cause of interferences in a wireless network. Secondly, the scheduling tool takes into account possible retransmissions due to interferences. A transmission is allowed to retry a few times, in hope of recovering from temporary noise spikes. Consequently, real-time requirements can still be met as long as the number of retransmissions is within a limit.

5. Modeling wireless process control system

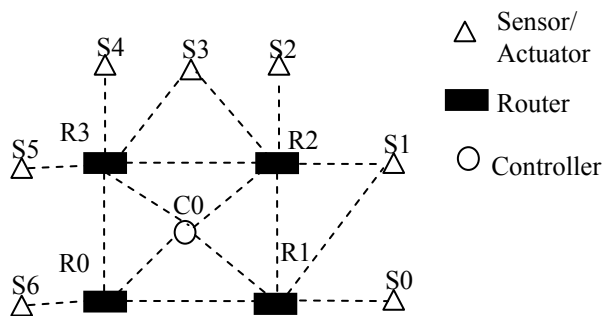


Figure 3. A simple wireless control system

In this section, we describe the details of translating a wireless process control configuration into a multi-processor scheduling problem solvable by the MSP.RTL tool. With MSP.RTL, we can produce a schedule for the control system so that most wireless interferences are avoided and a certain number of transmission failures are tolerated.

Figure 3 illustrates a wireless control system. This system is built on top of a ZigBee mesh network. Sensors/Actuators are end devices. When a sensor cannot

reach the controller directly, some intermediate routers have to be installed en-route to facilitate the communication. End devices cannot relay data from other devices. In addition, there can only be one controller in a network. These two requirements simplify our model: all control blocks will run in the controller; all input/output blocks will run in sensors/actuators. The sole function of the routers is to forward data packets.

Each function block can be mapped into a task with the corresponding period and run time. To schedule these tasks, a macro-cycle is first computed, during which each function block runs at least once.

Each node can be modeled as a processor. Embedded in each router is some type of routing algorithm. This routing algorithm runs in an asynchronous mode. Every time a router receives a packet, it searches the internal routing table and forwards the packet to the next hop. As the routing algorithm is not a periodic task, no periodic task is assigned to the routing process. Instead, a routing delay is used to simulate the routing process.

Until this point, this scheduling problem can be regarded as several independent scheduling problems on multiple processors. However, several aspects of wireless process control systems make this problem nontrivial:

- Communications between function blocks. Communications introduce many intricacies to this problem. First, two blocks that exchange data may have different periods if they belong to different loops. Secondly, a node cannot receive packets from multiple sources simultaneously. This is also the case when a node feeds the inputs to several receivers. Thirdly, there may be interferences between two communications. Communications are covered in Section 5.1.
- Two control loops may share one function block. In addition, the two control loops may have different periods. So a loop might only run certain instances of a block.
- A router may be shared by several control loops. A single control loop may traverse a router twice. In order to avoid interferences, the scheduler should assign the communications through that router to non-overlapping time slots.

5.1. Communications

A usual practice in real-time scheduling is to model communications as processes running on a bus. In a wireless control system, not all devices share a single bus. Two communications may take place concurrently if the transmitter and receiver of one communication are separated far enough from those of the other.

To capture this property of wireless communications, multiple buses are created for a wireless control system. In our model, all communications in a control loop are

considered running on a dedicated bus. As blocks in a control loop run sequentially in time, there would be no competition for the bus in a loop. We only need to consider the interferences between loops.

As routing processes are represented by routing delays, the two end points of a communication must be blocks. For example, in Figure 3, if a control block PID1 on controller C0 sends a packet to an output block AO1 on sensor S1 through the router R2, the communication is represented as PID1→R2→AO1.

If the two end blocks of a communication run at different rates, it can only be the case that the source block runs at a lower rate than the destination block, otherwise the receiver would be overrun. In this case, the source block can only send messages to certain instances of the destination block. Consider the communication AI1→PID1 in Figure 4. The rate of AI1 is 2HZ, while that of PID1 is 4HZ. Then, the first instance of AI1, AI1:1, can only send its data to either PID1:1 or PID1:2, but not both. This communication is represented as AI1:1→PID1:? in the process section. However, “?” can only be “1” or “2”, this restriction on “?” is listed in the constraint section of the input to MSP.RTL as follows:

$$\{ @(_AI1:1 \rightarrow PID1:?) \leq @(\^PID1:1) \text{ OR } @(_AI1:1 \rightarrow PID1:?) \leq @(\^PID1:2) \}$$

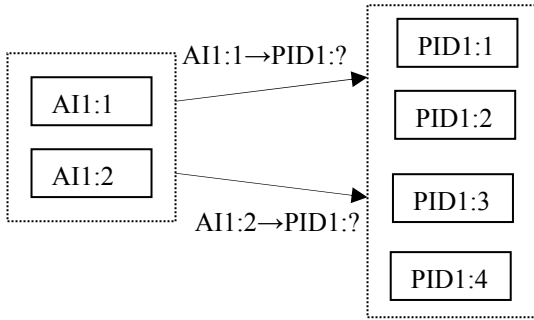


Figure 4: 2HZ sender and 4HZ receiver

5.2. Control loops

A control loop is comprised of block processes and communications that have to run in a sequential order. For example, in the control loop (AI1, AI1→PID1, PID1, PID1→AO1, AO1), AI1→PID1 can only start after the termination of AI1, and PID1 can not start until the end of AI1→PID1.

Some control loops contain feedbacks, as shown in Figure 2. Suppose the execution frequency of the loop is 2HZ, with a macro cycle of 100. Then the message sent by AO1:1 can only be used by PID1:2, and the message by AO1:2 would be received by the first instance of PID1 in the next macro cycle. The constraint for the last communication would be:

$$@(_AO1:2 \rightarrow PID1:?) \leq @(\^PID1:1) + 100$$

5.3. Shared routers and blocks

Routers and blocks may be shared by multiple control loops. In order to avoid interferences, we want to make the communications through shared routers/blocks run in non-overlapping time slots. Mutual exclusion can be easily described by an *OR* constraint in RTL. For example, if process A and B can not run at the same time, this requirement can be expressed as:

$$\{ @(_A) \leq @(\^B) \text{ OR } @(_B) \leq @(\^A) \}$$

Two communications without shared end nodes (blocks/routers) can still interfere with each other. However, the probability of interference is reduced dramatically. To address those infrequent interferences, we allow a communication process to retry a fixed number of times. As stated in Section 4, we assume that a transmission will succeed within a limited number of retries.

IEEE 802.15.4 [14], the PHY and MAC layer specification used in ZigBee, incorporates a variety of features to ensure reliable transmissions such as direct sequence spread spectrum (DSSS), quality assessment and receiver energy detection. With the 802.15.4 as the underlying MAC layer, we expect that the probability of transmission failures with several retries should be negligibly low and acceptable. In case such failure does happen, we can always resort to some emergency reporting mechanisms (such as alarms). It is the same case with traditional wireline control systems.

5.3. Scheduling and deployment

Once all loops are translated into MSP.RTL inputs, we run MSP.RTL to search for valid schedules that could then be deployed in the sensors/actuators and controllers in the networks. As long as our assumption about maximum retry is met, MSP.RTL guarantees that no deadline will be missed and no data will be lost.

6. Experiments and Results

In order to validate our approach, we use as benchmark a real plant configuration based on Foundation Fieldbus. In our experiments, this configuration is first converted into a ZigBee-based process control system. Then, the resulting wireless control configuration is translated into a multiprocessor scheduling problem to be solved by MSP.RTL, as discussed in Section 5.

6.1. The target wireless process control system

Due to physical limitations, a Fieldbus network can only support a certain number of devices. For the DeltaV digital automation systems [15], a maximum of 16 devices are allowed in each Fieldbus network. A process control system controls large plants by installing many Fieldbus networks. The summary for the real plant configuration is shown in Table 1.

Table 1. Summary of the real plant configuration

FieldBus Networks	25
Nodes (including controllers and devices)	158
Control Loops	125
Function Blocks	311
Communications	202

All control loops have the same period, 1000ms (1 second). The average run time of a control block is 10 ms, and the average run time of an input/output block is 45ms. The communication delay is 30ms.

Based on the plant configuration on Fieldbus, we construct the corresponding wireless control systems in different topologies. ZigBee supports three types of network topologies: star, cluster and mesh. In a star topology, all devices communicate with the central controller directly, without any relaying routers. As all devices on a Fieldbus network share a single bus, there is an inherent similarity between a Fieldbus network and a ZigBee network in a star topology. Thus we first map the configuration to a wireless control system based on ZigBee star networks. The mapping process is very trivial: all definitions of the wireline system can be literally copied to the wireless system with the exception of communications. As mentioned in Section 5.3, we allow a wireless communication to retry for a limited number of times. Thus the number of maximum retransmissions is specified as a parameter in the definition of a wireless control system. Other than this parameter, there exists a one-to-one mapping between the Fieldbus plant configuration and the ZigBee configuration. The resulting wireless control system is saved in a data file which lists the devices in the network, the function blocks on each device and the composition of each control loop. The definition of the derived wireless system can be found in [16], the full version of this paper.

In addition to ZigBee star networks, we also carry out another experiment on ZigBee mesh networks, where end devices are connected to the controller via routers.

6.2. Experimental Results

In order to schedule control loops in the wireless control system, we wrote a parser to translate the configuration file to a scheduling problem definition in RTL format. The details of the translation process were discussed in Section 5.

The parser and MSP.RTL are written in Java with *Netbeans IDE 5.0*. The experiments are carried out on a *Dell Latitude C400* laptop running Windows XP with Intel Pentium III 1.2GHZ CPU and 512MB memory. The run time is averaged over 10 runs of each program.

As the first step, the definition of the wireless control system is fed to the parser. Table 2 summarizes the scheduling problem resulting from the wireless system. Of the 283 processors, 125 processors are virtual buses created for communications, which is more than the number of networks (25). For networks in a star topology, we could be better off creating only one bus for each network. For networks in cluster/mesh networks, there could be multiple buses associated with a network. In order to ensure that our parser works for networks in all three topologies, we choose to create a bus for each loop, as described in Section 5.1. An experiment on ZigBee mesh networks is presented at the end of this section.

Table 2. The resulting scheduling problem

Processors	283
Processes	512
Constraints	1190

Table 3. Schedule results for different retries

Max Retries	Schedulability	Run Time(s)
2	Yes	7
3	Yes	8
4	Yes	9
5	No	0

The run time of the parser is always less than 1.0 second, which is negligible for design purposes. In the second step, we run MSP.RTL on the output created by the parser. We vary the maximum number of retransmissions from 2 to 5. The result is shown in Table 3.

As shown in Table 3, the run time goes up with the increase in the maximum number of retries. When the number of retries is incremented, each communication takes a longer time, which causes MSP.RTL to allocate more time to each communication and thus to spend more

time finding a feasible schedule. When the maximum retries reaches 5, MSP.RTL promptly reports that this problem is unschedulable. It is worth noting that the run time of MSP.RTL is more related to the internal structure of a wireless control system than the size of the system. As mentioned in Section 6.1, a control system is composed of several separate networks. The schedule of each network is independent from those of other networks. For MSP.RTL, if the size of the control system is increased by adding more networks to it, the running time is increased at most linearly. However, if the size of a system is kept intact and only one network becomes more complex, MSP.RTL may take much longer time to find a schedule. In an extreme case, if a network is unschedulable by MSP.RTL, the whole control system would be unschedulable.

To validate the schedule independently, we implement a verifier which checks if the schedule is consistent with all constraints in the problem. The schedule created by MSP.RTL passes the verifier successfully.

To test the parser and MSP.RTL in a more general environment, we map a network in the real plant configuration to the ZigBee mesh network shown in Figure 3. The definition of the corresponding wireless system is described in [16]. The routing delay is set to 5ms and the maximum number of retries is 2. From the definition, it is obvious that R3 is heavily loaded: control loop 42TI12019, 42XVS12039 and 42XVS12041 traverse it. Our MSP.RTL tool is able to produce a schedule for the six loops within 1.0 second, and keep the communications through router R3 mutually exclusive.

7. Conclusion and Future Work

In this paper, we investigate the feasibility of process control over wireless networks from the perspective of real-time multiprocessor scheduling. Based on several reasonable assumptions, we show how to map a traditional Fieldbus based control configuration to a wireless system composed of ZigBee star networks. We discuss the details of translating a wireless process control systems into a multi-processor scheduling problem. Then we apply a scheduler synthesis tool, MSP.RTL to solve the scheduling problem. The resulting offline schedule is deployed in the sensor network to achieve real-time control. Simulation results on real plant configuration data make us believe that wireless process control is worth pursuing.

In addition to periodic loops, there are also sporadic events in a control system, such as alarms and events. We plan to take those sporadic tasks into account in future work. Another avenue of future work is to investigate the impact of a sensor network's multi-path property on our method.

8. References

- [1] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks, *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001
- [2] D. Chen, M. Nixon, T. Aneweer, R. Shepard, and A. K. Mok, Middleware for Wireless Process Control Systems. *Architectures for Cooperative Embedded Real-Time Systems Workshop*, December 2004
- [3] S. Ganeriwal, R. Kumar, M. B. Srivastava, Timing-sync protocol for sensor networks. *ACM Conference on Embedded Networked Sensor Systems (SENSYS 2003)*, November 2003
- [4] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, *The 6th Annual International Conference on Mobile Computing and Networks (MobiCom 2000)*, Boston, MA, August 2000
- [5] Q. Li and D. Rus. Global Clock Synchronization in Sensor Networks. *IEEE Infocom 2004*, Hong Kong, China, March 2004
- [6] T. Liu, C. Sadler, P. Zhang and M. Martonosi. Implementing Software on Resource-Constrained Mobile Sensors: Experiences with Impala and ZebraNet. *The Second International Conference on Mobile Systems, Applications, and Services (MobiSYS'04)*, June 2004
- [7] A. Mok, D. Tsou, and R. Rooij. The MSP.RTL Real-Time Scheduler Synthesis Tool. *The 17th IEEE Real-Time Symposium (RTSS '96)*, Washington D.C., December 1996
- [8] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. *Mobile Computing and Networking*, 2001
- [9] Foundation Fieldbus standard, <http://www.Fieldbus.org>
- [10] Profibus standard, <http://www.profibus.org/>
- [11] WINA, <http://www.wina.org/>
- [12] ZigBee Alliance, <http://www.zigbee.org/>
- [13] Chipcon Products, <http://www.chipcon.com/>
- [14] IEEE 802.15.4 Task Group, <http://www.ieee802.org/15/pub/TG4.html>
- [15] DeltaV Home page, <http://www.easydeltav.com/>
- [16] J. Song, A. Mok, D. Chen, and M. Nixon. Using Real-Time Logic Synthesis Tool to Achieve Process Control over Wireless Sensor Networks. UTCS Technical Report, <http://www.cs.utexas.edu/~sjp/publications/mspwn.pdf>, Austin, TX, March 2006

9. Acknowledgement

This research reported here is supported partially by a grant from the Office of Naval Research under contract number N00014-03-1-0705.