Object tracking in a multi sensor network

Sebastiaan de Vlaam

The Hague, August 17, 2004

Delft University of Technology
Computer Engineering, Parallel and Distributed Systems

TNO Physics and Electronics Laboratory
Smart Sensor Solutions, Networked Embedded Systems

Abstract

Wireless Sensor Networks (WSN) are networks of small "sensor nodes", small computing devices with wireless communication abilities and sensors. The last few years, there has been a lot of research on sensor networks and their applications. This report presents such an application of sensor networks. The objective is to design and implement an object tracking system using a wireless sensor network. This application is able to detect and track objects, and report information about these objects to a central base station.

A passive infrared (PIR) sensor is chosen to detect objects, especially humans. This PIR sensor is connected to a Mica2 node, a commercially available sensor node. The PIR sensor is able to detect humans and provide an estimation of the direction of movement. A PC with a node programmer is used as a base station to display and gather information from the sensor network.

The software for a node is divided into different modules, each to perform a specific task. There are modules for the PIR sensor, detecting movement, sending messages (multicast), and a top-level module for the tracking algorithm. These modules are written in nesC for the TinyOS operating system, that runs on the Mica2 nodes. All modules have clearly specified bidirectional interfaces that describe the provided functions.

Experiments with the object tracking system show that the system can track an object, calculate its speed, and report this information to a base station. The speed calculation is relatively accurate, but can be influenced by the orientation of the sensor.

The object tracking system using multiple sensors provides a good foundation for a security application and shows the possibilities of sensor networks in this field. The usability of the system can be improved by incorporating localization and more configuration possibilities. This will be added to the system, before it is used in a demonstration for the Ministry of Defense in October 2004.

Preface

This report describes the project done for my thesis for the Master of Science degree in Computer Engineering at Delft University of Technology. This project was performed under the supervision of dr. ir. M.G. Maris at TNO-FEL in The Hague and dr. K.G. Langendoen at TU Delft. The graduation committee is composed of:

prof. dr. ir. H.J. Sips	TU Delft - Parallel and Distributed Systems
dr. K.G. Langendoen	TU Delft - Parallel and Distributed Systems
dr. ir. A.J. van Genderen	TU Delft - Computer Engineering
dr. ir. M.G. Maris	TNO Physics and Electronics Laboratory

Acknowledgments

I would like to thank TNO for giving me the opportunity to work within their organization and providing me with a master project with a practical application. In particular thanks to Marinus Maris and Maarten Ditzel for their support and comments. Also I would like to thank Piet l' Ami for constructing the (tiny) connectors for the sensors. Of course I would like to thank Koen Langendoen for his supervision on my work for this thesis.
Last but certainly not least I would like to thank my girlfriend Hanneke Siegers for always wanting to listen to me.

Sebastiaan de Vlaam
August 17, 2004

Contents

Chapter 1

Introduction

This chapter gives a short introduction to the project done for my Master in Computer Engineering. It provides the context in which this project has been done, an introduction to the organization in which the project has been performed and a general introduction to sensor networks.

1.1 Project context

Object tracking is being investigated as part of a research program at TNO-FEL to demonstrate the usability and possibilities of sensor networks to the Ministry of Defense. This project is inspired by the "Compound Security Demonstrator" project [9] and provides some important extensions as well as improvements.

Protection of personnel has a high priority in peacekeeping missions. When a military unit establishes a base (compound) they want to set up a security system as quickly as possible. At the moment the royal army has different sensor systems, including footstep detectors, a battlefield radar and trip wires. These systems have several disadvantages including single points of failure and (semi) constant personnel attention. So there is a need for a system that is easy to deploy and does not require an existing infrastructure or constant checkup by humans.

1.2 About TNO

TNO (Netherlands Organization of Applied Scientific Research) is a knowledge organization for companies, government and social organizations. Its main objectives are research and the practical applications of this research. TNO has around 5400 employees. Services provided by TNO include research on demand, advise, testing and certifying products, and independent quality control.

TNO Physics and Electronics Laboratory (TNO-FEL) provides leading edge services and products in the field of military (defense), public safety, ICT, transport and logistics, aerospace- and electronic systems. At TNO-FEL employs 550 people. The project described in this report has been executed within the division Smart Sensor Solutions in the Networked Embedded Systems (NES) group.

1.3 Sensor networks

Wireless Sensor Networks (WSN) are networks of small "sensor nodes" consisting of a microcontroller, a radio front-end, a power supply and one or more sensors enabling deeply embedded systems capable of sensing the physical environment. The development of these sensor networks is fueled by the advances in computer chip miniaturization and circuit design allowing for small devices with efficient wireless communication equipment. The last few years there has been a lot of research on sensor networks. A few of the challenges faced in the design of sensor networks include localization and ad-hoc networking. The goal is to deploy a network of hundreds or thousands of nodes that organize themselves into a network and start sensing the environment. Examples of applications of sensor networks include environmental control, surveillance and tagging mobile items [4].

1.4 Report outline

This report has the following structure. Chapter 2 describes the objective and requirements for this project. Chapter 3 describes and motivates the choices for the hardware and software used in object tracking. Chapter 4 provides an overview of how the system should look and identifies the main components. In Chapter 5 a design is made for the application running on the nodes and a design is presented for software running on a PC that serves as the base station. Chapter 6 covers the details of the implementation of the software. Chapter 7 describes the experiments and their results. The experiments were performed both with the entire system as well as some test systems. Finally, Chapter 8 concludes with final remarks and suggestions for future research.

Chapter 2

Objective and requirements

This chapter describes the objective and the requirements for this project. The requirements do not include specific requirements for the use in a military environment, for example, robust packaging, as they can always be added afterwards.

2.1 Objective

The objective of this project is to design and implement an object tracking system using a wireless sensor network. The system should report the location of the object and possibly other information about a detected object, such as speed and direction. Information about the detected object should be acquired and shared among the sensor nodes using the wireless communication possibilities of the nodes. This shared information will be used to generate precise information about the object and to improve the reliability of the object detection and classification.

2.2 Requirements

- Detect physical intrusion within a specified area.
  The system should primarily detect humans. The area is bounded by the sensing range of the sensors.
- Determine the location, velocity and possibly the size of a detected object.
  Using the sensor information, as much information as possible should be reported about the detected object.
- Sensor nodes share their sensor information to improve the reliability of the individual sensor readings.
- Report to a single base node (in sensor networks also known as the "sink") and receive messages from this base node.

All nodes should be able to send a message to the base node and receive messages from the base node regardless of the network topology.

- Detect and handle disabled/removed nodes.
  The system should be able to detect the loss of one or more nodes and keep functioning as reliable as possible.

- Extend the network with additional sensor nodes.
  New sensor nodes should be able to join the sensor network without additional configuration.

- Low power operation.
  The entire system of sensor nodes should use the least possible energy using the sleep and/or power-down capabilities of the sensor nodes.

- The system should respond within half of the time needed for a person to go from one sensor to the next. Thus within $\frac{1}{2} \times \frac{d}{v}$ seconds, where d is the distance between two sensors and v the speed of person. The average walking speed of a person is around $1,4\ m/s$ (or $5\ km/h$).

Chapter 3

Hardware and software choices

In this chapter the hardware and software used for the project is described. If necessary, choices for a certain part of hardware and/or software are motivated.

3.1 Nodes

The next sections cover the hardware and software for the wireless sensor nodes.

3.1.1 Hardware

As hardware for the wireless sensor network, the *Mica2* [5] platform is used. The Mica2 is a module for low-power, wireless sensor networks. It was designed at the Berkeley University of California [1] and is manufactured and distributed by Crossbow [2]. A Mica2 node is shown in Figure 3.1.

Figure 3.1: Mica2 wireless sensor node.

The Mica2 is based on an ATmega128L 16MHz low-power microcontroller, it contains 128K bytes of program flash memory, 512K bytes of measurement (serial) flash and 4K bytes of EEPROM.

The Mica2 platform was chosen because of the availability at TNO-FEL. The TU Delft also does some research on the Mica2 nodes. The knowledge of the Mica2 platform within TNO-FEL makes it possible to incorporate obtained knowledge into other projects more easily. The TinyOS-Berkeley/Crossbow nodes are worldwide one of the most used platforms for wireless sensor networks, which is interesting from a commercial point of view.

The Mica2 nodes are programmed using a MIB510 serial programmer provided by Crossbow. Besides programming the nodes, this programmer (together with a node) can be used to communicate between a PC and a sensor network.

3.1.2 Software

TinyOS [7] runs on the Mica2 platform. TinyOS is an event-driven open-source operating system for wireless (embedded) sensor networks. The TinyOS-Berkeley/Crossbow combination is used by over 500 research groups. Many individuals and research groups contribute to the code and standards of TinyOS via its sourceforge website [6].

TinyOS is written in nesC, a programming language with a C-like syntax for programming network embedded systems. NesC applications consist of one or more *components* linked together to form an executable. A component provides and uses interfaces. Interfaces declare a set of functions called *commands* that the provider of the interface must implement and another set of functions called *events* that the user of the interface must implement. NesC has two types of components: *modules* and *configurations*. Modules implement one or more interfaces. Configurations are used to assemble other components together, connecting interfaces to their implementation. This is called *'wiring'* [8].

The entire TinyOS operating system, including all libraries, is written in nesC. All the software, running on the Mica2, for this project is also written in nesC.

3.2 Programming environment

The TinyOS tools are available for Linux/Unix and Windows (under Cygwin) and contain various tools:

- NesC compiler,
- AVR compiler and utilities,

- Java SDK and Java COMM,
- A sensor network simulator.

TinyOS provides tools to convert TinyOS messages to Java classes, so the messages can be used in Java programs. This way Java can be used to create programs to communicate with the nodes, using a serial connection via the programmer and a dedicated node on the programmer.

The TinyOS tools also come with a Java program called the "Serial Forwarder", which can provide communication with the sensor network (using a programmer and node) via a TCP/IP network. This tool can be used to link any program to the sensor network.

3.3 Sensors

In this section the sensor chosen for this project is described.

3.3.1 Types of sensors

For a system to be used for intrusion detection, a sensor should be used that is capable of detecting (moving) objects. Other requirements for a sensor are:

- low-power operation,
- processing sensor data should not require too much processing power,
- processing sensor data should not require too much processing time,
- reasonable size and cost,
- reliable, meaning it should not give false positive or negative readings.

Considering the requirements different types of sensors can be used. For object tracking the following sensors can be considered: accelerometer (seismic), ultrasound (ultrasonic) and infrared (thermal) [3].

Sensor	**Low-power**	**Processing**	**Reliability**	**Size & cost**
Accelerometer	yes	low	low	low
Ultrasound	no	high	high	high
Infrared	yes	low	medium	low

Table 3.1: Properties of different types of sensors.

3.3.2 Sensor choice

Considering the information in Table 3.1, the ultrasound sensor is excluded as a sensor with low-power properties is required. Comparing the accelerometer and the infrared sensor, the infrared sensor has better detection properties for movement. Thus the sensor used in this project will be a thermal sensor, more precisely a passive infrared sensor (PIR).
The choice for the infrared sensor is also based on the fact that a PIR is already used in the *Compound Security Demonstrator (CSD)* [9] which provided good results for direct motion detection. Another advantage is that the analog output signal of a PIR sensor can give an indication of the direction of movement (see Figure 3.3 and Section 5.1.3).

Operation of PIR sensors

Objects that generate heat also generate infrared radiation. The wavelength of infrared radiation is longer than the wavelength of visible light. It can not be seen, but it can be detected by a pyroelectric sensor. This sensor is made of a crystalline material that generates an electric charge when exposed to infrared radiation (i.e. heat). The amount of charge is proportional to the amount of radiation. Pyroelectric elements are sensitive to radiation over a wide range, therefore a filter window is placed before the sensor to limit incoming radiation to the 8 to 14 μm range, which includes the infrared radiation from human bodies.

Usually a PIR sensor package contains two elements next to each other and the output of the PIR sensor is based on the difference between the two elements. This cancels out signals caused by vibration, temperature changes and sunlight. A person passing the sensor will first activate one element and then the other, which gives a positive or negative difference between the elements depending on which element is activated first. See Figure 3.2 and Figure 3.3.

The PIR sensor used for this project is an all-in-one package, containing a PIR sensor, some circuitry (including an amplifier) and a Fresnel lens. This sensor is available from Conrad, type number 172500. See Figure 3.4.
The advantage of the selected PIR sensor is the possibility to use the analog output. Another advantage is that this sensor can easily be mounted on a wireless sensor node without increasing the total size dramatically.

3.3.3 Sensor connection

The PIR sensor has to be connected to the Mica2 node. The Mica2 has a 51-pins connector for attaching sensor boards. The pin assignment of this connector is described in the documentation available from Crossbow [2].

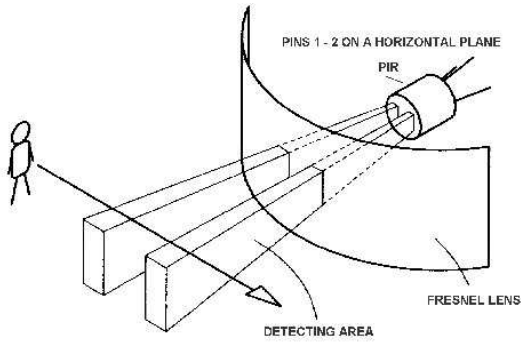

Figure 3.2: Illustration of movement detection with a PIR sensor.

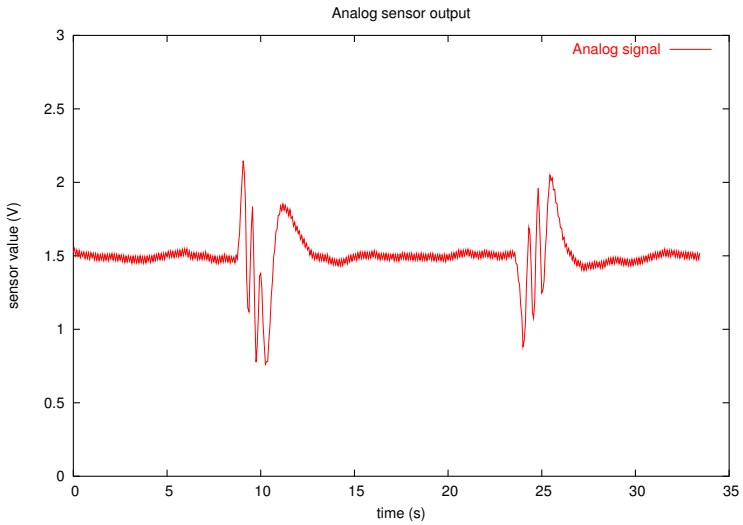

Figure 3.3: Analog output signal representing movement in both directions.

Figure 3.4: PIR sensor.

The PIR can interface directly with the extension connector without additional circuitry. The pin conversion is shown in Table 3.2.

PIR sensor	Mica2 Connector
1 (ANA)	42 (ADC1)
2 (GND)	1 or 51 (GND)
4 (OUT)	43 (ADC2)
5 (GND)	1 or 51 (GND)
6 (VCC)	2 or 50 (VCC)

Table 3.2: Pin conversion from PIR sensor to Mica2 connector.

Using this pin assignment a connector is made to connect a PIR sensor to the Mica2 node. This connector is shown in Figure 3.5 (including a PIR sensor).

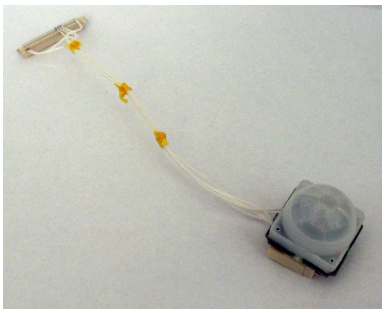

Figure 3.5: Mica2 connector for the PIR sensor, including the PIR sensor.

3.3.4 Sensor operation and output

A PIR sensor has an input voltage (V_{cc}) ranging from 3 to 12 V using 1,4 mA. The 3 V can easily be drawn from the same batteries that operate the node (2xAA), which excludes the need for an external power source for the sensor. This also prevents another increase in size. The sensor has three outputs, an analog (data) output, a reference voltage and a digital output. The digital output indicates if movement is being detected, the reference voltage is approximately half the supply voltage $\frac{V_{cc}}{2}$, and the analog output gives a (theoretical) value between 0 and V_{cc}. The reference voltage is not connected and therefore cannot be used.

The sensor is connected to the Mica2 as explained above. The sensor is powered directly via the 51-pins connector and thus powered whenever the node power is enabled. The sensor requires a startup period of at least 45 seconds, regardless of the sensor has been turned on before. This behavior

is the result of the charging of capacitors and the circuit. Because the amplifiers in the sensor act as a diode when the power is turned off, the entire circuit has to be recharged when the power on the sensor has been turned off.

Chapter 4

System

In this chapter the total system setup is presented, including how the system looks and operates.

4.1 Components

The entire intrusion detection system consists of the following components:

- A number of sensor nodes with a passive infrared sensor, as shown in Figure 4.1. These nodes will be running the software described in 3.1.2. Of course for outdoor use the node and sensor have to be placed in a robust packaging. Also, there has to be a way to easily place the nodes, for example attaching them to a building or placing them on special posts or fences.
- A base node, which communicates with a PC through the programmer and a serial cable. See Figure 4.2 for a picture of the MIB510 programmer.
- A PC with the programmer connected to it, functioning as base station for the system. The PC serves as an actuator in this system reporting information obtained from the sensor network to the user. The PC can be replaced by another actuator, for example a camera or an alarm.

4.2 Placement

To detect movement, there is a number of considerations for the placement of the entire network and the individual nodes. PIR sensors need a direct line of sight for detection and are not omnidirectional, so the orientation of the sensor has to be considered. Depending on the network topology, the sensors all have to have the same orientation. For all nodes to be able to reach the base station, it has to be placed centrally. See Figure 4.3 and 4.4 for a schematic view of the system.

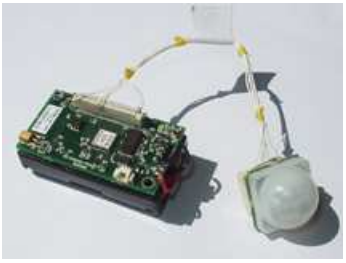

Figure 4.1: Mica2 node with a PIR sensor connected to it.

Figure 4.2: MIB510 serial programmer.

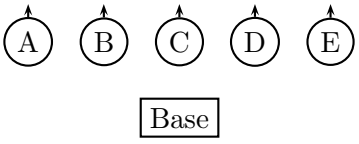

Figure 4.3: Line topology, the arrows indicate the sensor heading.

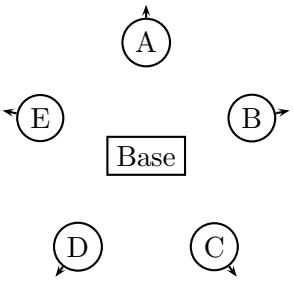

Figure 4.4: Circle topology, the arrows indicate the sensor heading.

4.3 Assumptions

To be able to track a moving object some assumptions are made about the movement of the object and the properties of the detection and tracking. Some assumptions are consequences of the chosen solution and some are consequences of the (at the moment) limited capabilities of sensor networks.

- If an object is detected by a node and after a certain time period an object is detected by another node, it is presumed to be the same object. This assumption is made because objects do not carry any form of identification, nor can different objects be distinguished. This assumption is a consequence of the chosen solution. A PIR sensor cannot distinguish between different objects. Multiple heterogeneous sensors on the same node (sensor fusion) can be used to distinguish between different objects.

- Each node "knows" who its neighboring nodes are and knows their relative position to its own position (either left or right). Neighbors in this case are the nodes directly adjacent to a node and not, as usually in sensor networks, all nodes within radio range. The sensing range of the PIR sensor is set to about 4 m, which is considerably less than the radio range. So, in the interpretation using radio range, a node has a lot of neighbors. The information about neighboring nodes is initially stored in the software. Ideally a node determines its own (relative) position using localization.

- The nodes form a network that can detect movement in one dimension, see Figure 4.3 and 4.4 for possible topologies. This assumption is also a consequence of the chosen solution. By limiting the detection field to one dimension, all nodes can obtain the information about the object (direction, speed) with only relative information from the neighboring nodes (left or right, distance). Extending the detection field to two dimension, for example by placing the nodes in a grid, requires more position information. The position can then be specified by a coordinate system, which requires some (common) origin.

- The nodes are distributed in such a way that the distance between the nodes is always the same between neighboring nodes. The distance is used for speed calculation. Using localization methods, a node can determine its own distance to its neighbors.

Chapter 5

Software design

This chapter presents the software design for both the software running on the nodes and the software running on a PC. First the structure of the system is described and after that the individual modules are explained.

5.1 Node software

This section describes the design of the software that is running on the nodes.

5.1.1 Modules

The software for a node is separated into a number of modules, each with its own functionality. The relation between the modules is shown in Figure 5.1. Modules shown with a dashed line (Leds, GenericComm, ADC) are provided by TinyOS. The node software consists of the following modules.

- **Infrared**
  This module provides an interface to the ADC (Analog Digital Converter). It can read the analog and digital values from the PIR sensor. It is used by the Detect module.
- **Detect**
  The Detect module signals an event to the Tracking module upon movement detection, including a direction indication. It is also responsible for resetting the sensor state. All sensing parameters are specified in this module.
- **Multicast**
  This module provides the functionality to send one message to any number of nodes. It is used by the Tracking module to communicate with other nodes.

- **Tracking**
  This is the main module containing the functionality for communication between nodes and reporting events to the base station. The configuration of this component wires all the components mentioned in this design, including the necessary libraries for communication. It uses the Detect module to get a signal upon detection of movement and the Multicast module for sending messages to neighboring nodes.

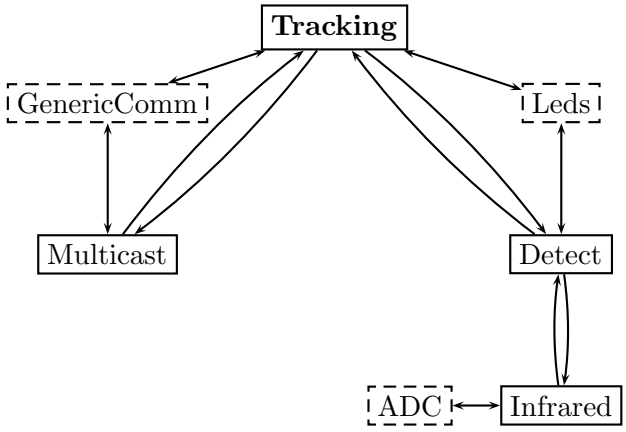

Figure 5.1: Tracking application modules and their relations.

5.1.2 Infrared

The Infrared module provides an interface to the ADC. Table 3.2 shows to which ADC channels the PIR sensor is connected. The analog output of the PIR sensor is actually the only output that is used for this application, but the digital channel is provided for future use and/or applications.

TinyOS provides an interface to the ADC, which will be used in the Infrared module. A separate interface is created for accessing the infrared sensor to shield the user from being confronted with irrelevant details of accessing and initializing the ADC. When wiring (see Section 3.1.2) the ADC interface, a direct reference to the used ADC channels has to be specified. Also an interface for control and initialization of the ADC is wired.

The Infrared module provides commands to read from the different infrared outputs (i.e. analog and digital) and provides an event that is signaled when a value is read from the ADC. The sampling rate is not specified in this module, but in the Detect module. The Infrared module just reads a sample from the ADC and signals an event containing the sample data.

5.1.3 Detect

The Detect module uses the infrared interface to detect movement and the direction of the movement. It provides a command to start the detection and provides an event, which is signaled upon motion detection, that indicates the detected direction. As mentioned previously, the Detect module also specifies the sampling frequency for the sensor.

The direction of movement can be determined from the analog sensor data. Approaching the sensor from one direction gives a positive peak in the signal and approaching from the other direction gives a negative peak (See Figure 3.3 and Section 3.3.2). To prevent that noise is mistaken for movement, a threshold T is set. The sensor data must exceed this threshold to signal an event, which indicates that movement is detected. This threshold is relative to the data average and is symmetrical. This means that the same threshold is used for both directions, using $average + T$ for one direction and $average - T$ for the other, see Figure 5.2

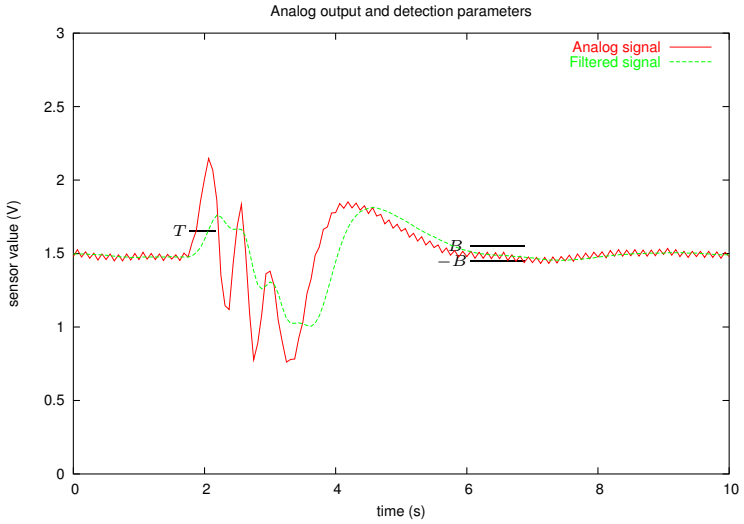

Figure 5.2: Typical sensor measurement and parameters for movement detection.

When a person is passing the sensor, the threshold is being exceeded frequently, but the sensor is still detecting the movement of one person. Thus, to detect the end of one movement, the sensor value has to be between a relative boundary B for a number of consecutive samples (see Figure 5.2).

The raw output signal of the sensor is not smooth (Figure 5.2, solid line), because is is dependent of the fluctuating supply voltage. These fluctuations

are caused by other operations on the Mica2 requiring (non-constant) power, for example radio communications. Therefore, the data is filtered using a moving average filter with a window size of 10 samples (see the green line in Figure 5.2) to smooth the data.

It is empirically determined that 8 samples per second from the PIR sensor provides enough data to detect motion with direction. Lower sampling rates increase the possibility to miss objects or misinterpret the direction information. Higher sampling rates consume unnecessary (processing) power, and experience more influence from other operations on the Mica2. The detection algorithm is given below in Algorithm 1.

```
input: threshold T,
       boundary B,
       sample frequency f,
       samples between boundary n
begin
    i ← 0
    detecting ← FALSE
    while timePassed(1/f) do
        call readAnalog()
        wait readDone(sample)
        filter(sample)
        if sample < T or sample > T then
            signal detected(sample < T)
            detecting ← TRUE
        end
        else
            if sample > −B and sample < B then
                i ← i + 1
            end
            else
                i ← 0
            end
            if i >= n then
                detecting ← FALSE
            end
        end
    end
end
```

Algorithm 1: Detection algorithm.

5.1.4 Multicast

The Multicast module provides a way to send a message to multiple, but not all, nodes. It is called multicast, but actually it implements the repeated transmission of the same message to multiple nodes. This mechanism is wrapped in a single interface to shield the details of sending a message multiple times from the user. The algorithm for this module is shown in Algorithm 2.

input : Array of node addresses N
Number of nodes in the array n
Message msg
ouput: Event signaling message send successful or unsuccessful

begin
  for $i \leftarrow 0$ ***to*** $n-1$ **do**
    `send(`msg`)` to $N[i]$
    wait $success \leftarrow$ *`sendDone()`*
    if *!success* **then** break
  end
  signal *`multicastDone(`*success*`)`*
end

Algorithm 2: Multicast algorithm.

The main advantage of the Multicast module is to improve readability of a program as only one call is needed instead of a mechanism to handle multiple sending of the same message. With point-to-point messages, it is also easier to add mechanisms such as acknowledgment and CRC (Cyclic Redundancy Check).

5.1.5 Tracking

The Tracking module is the top-module for target tracking. Ultimately, it provides an estimation of the position and the speed of a moving object (person). To realize this, the nodes have to cooperate and communicate. Communication is done using both the communication capabilities of TinyOS directly (for communication to the base station) and the multicast interface for communicating with a node's neighbors.

The Tracking module performs two (closely related) functions: coordination (using communication) between nodes and communication to the base station. These functions are closely related because the same information, i.e. the same message, is sent to neighboring nodes and the base station, respectively. This message will also contain the information needed to determine which node sends information to the base station.

Object tracking

Upon detection of an object the Detect module signals an event which is implemented by the Tracking module (see Section 3.1.2). This event contains directional information obtained from the local sensor. The directional information is processed differently depending on the state of the tracking: tracking an already detected object or starting to track a new object.

If a node starts tracking an object it assigns a "target id" to that object. Also the directional information from the Detect module is assumed to be the actual direction of the object. However, a flag is set indicating that the acquired directional information has not yet been confirmed by other nodes.

If a node detects an object and it is in the state of still tracking a (previously) detected object, the direction of movement is determined from the relative position of the neighboring node that detected the object earlier, rather than using the local sensor information.

After the direction of an object is determined, a message is sent to both neighbors of a node, containing the following information:

- the node address of the sending (source) node,
- the determined direction,
- a flag indicating whether the direction has been confirmed by other nodes,
- the determined speed (if possible),
- a target id of the object that is being tracked.

This message is sent to both the left and the right neighbor to prevent the situation that if the local sensor information is wrong, the second node, which detects the same object, sees it as a new target. See figure 5.3, the red arrow indicates the direction detected by node B, which is incorrect. If node B assumes the detected direction is correct, it would only need to send a message to node A. Node C will then detect the object as being a new target (which it is not). Therefore the message is send in to both neighbors. The messages to to the neighbors are sent in a wraparound fashion to support ring topologies (see Figure 5.4).

Upon receiving a message from a neighboring node, a timer is started with two purposes: to specify a timeout period in which a node waits to detect the target it has received information about, and to determine the time t it

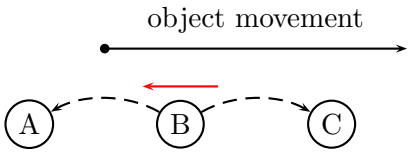

Figure 5.3: Detection of the wrong direction.

took the object to travel the distance D between nodes. Using this timer a node can calculate the speed of the object using the simple formula:

$$v = \frac{D}{t} \tag{5.1}$$

where v is the calculated speed in m/s.

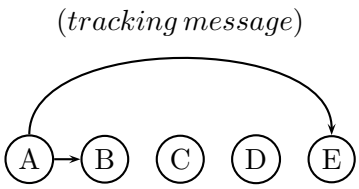

Figure 5.4: Direction of message sending.

Base station reporting

To use the sensor network for target tracking, the obtained information about a tracked object needs to be communicated to a base station (PC or other actuator, see Section 4.1). The Tracking module implements a simple scheme to determine when to send information to the base station. In the software, a number n can be specified, which indicates which node should send a message to the base station. This means that every nth node that detects the target sends the tracking information to the base station (for example, with $n = 2$, the second, fourth, sixth, etc.).

Nodes have to communicate to determine which one is the nth node. To do this a number called *node count* is included in the message sent to neighboring nodes upon detection of movement. If $modulo(node\,count, n) = 0$ then the same message that is sent to both neighbors is also sent to the base station, which then knows the position of the object. For $n > 1$ the message contains speed information so the position of the detected object can be extrapolated in time.

The choice for the value of n can depend on a few factors:

- the distance between the nodes,

- the (expected) speed of the object(s),
- the required amount of feedback from the network.

5.2 Base station software

As mentioned in Section 4.1, the base station consists of a base node to communicate (wirelessly) with the sensor network and a PC. The base node runs a simple message forwarding program and the PC runs software to interpret and process messages from the network.

5.2.1 Base node

The base node's only task is to forward messages from the sensor network to the PC. This is done by connecting the base node to the PC using a programmer. The TinyOS package comes with a program called TOSBase (TinyOS Base) to intercept all packets from the network. Because only messages sent to the base station (address) needs to be received, a slightly altered version of TOSBase has been used. This version still intercepts all packets, but eventually drops the ones not meant for the base station. An advantage of using TOSBase is that TOSBase is bidirectional. It can also send messages received from the PC to the sensor network.

5.2.2 PC

At the moment the PC program consists of a simple Java application that shows the received messages in human readable form. Using the structure that defines a message, a Java class is generated. This Java class, together with the TinyOS java tools, provides an easy way to display the information contained in a message. See Figure 5.5 for a example of a displayed message.

```
<TrackingMsg>
  [sender=0]
  [source=2]
  [direction=0]
  [speed=1.090909]
  [indication=0]
  [node_count=3]
  [target=407]
```

Figure 5.5: Message displayed by the PC program.

Chapter 6

Software implementation

In this chapter some consequences of the design presented in Chapter 5 are discussed.

6.1 Node program modules

The next sections give the specification of the interfaces of the modules introduced in Section 5.1.1 and give implementation details of the different modules.

6.1.1 Infrared

As seen in Table 3.2, ADC channel 1 is used for the analog output of a PIR sensor and ADC channel 2 is used for the digital output. All events from the ADC channels are mapped onto one event in the infrared module. The commands *readAnalog* and *readDigital* are specializations of the ADC command *getData*. They operate on the specified ADC channels.

The definition of the Infrared interface is as follows:

```
interface Infrared {
   command result_t readAnalog();
   command result_t readDigital();
   async event result_t readDone(uint16_t sample);
}
```

6.1.2 Detect

The implementation of the detection module follows Algorithm 1. The state *detecting* specifies if an object has been detected, this prevents false signals. The while loop specified in Algorithm 1 is implemented by a timer, where the body of the while loop is executed within an event that is signaled each time the timer fires. Experimenting with the different parameters shows that the

values in Table 6.1 provide a good detection and a reasonable response time between detections (i.e. the time after which a new object can be detected).

Parameter	Variable	Value
Average	$AVERAGE$	512
Detection threshold	T	30
Rest state boundary	B	10
Sample frequency	f	$\frac{1024}{64}$
Samples between boundary	n	5

Table 6.1: Detection parameters.

The definition of the Detect interface is as follows:

```
interface Detect {
   command result_t startDetection();
   async event result_t detected(bool direction);
   event result_t reset();
}
```

6.1.3 Multicast

The Multicast module uses the TinyOS interface SendMsg for sending messages to other nodes. The SendMsg interface provides a parameterized interface (interface that can provide multiple instances), with the parameter being a handler id for the message type. To generalize the multicast module it should also be defined as a parameterized interface with the parameter being the message type. As yet there is no way to pass parameters to another component within a configuration. In this case a parameter needs to be passed from the provided interface (i.e. multicast) to a used component (i.e. SendMsg). This problem is solved by including a header file in which the message type can be specified.

Consistent with the SendMsg interface, the multicast module implements a command *send* with an array of node addresses (instead of a single address in SendMsg), the number of nodes in the array, the size of the message and a pointer to the message which has to be sent as parameters. The module sends the message to the first node in the array and waits for the *sendDone* event, it then sends the (same) message to the next node in the array and so on. Upon successful sending the message to all nodes it signals its own *sendDone* event.

The definition of the Multicast interface is as follows:

```
interface Multicast {
   command result_t send(uint16_t *addresses, uint8_t num_nodes,
                         uint8_t size, TOS_MsgPtr msg);
```

```
    event result_t sendDone(TOS_MsgPtr msg, result_t success);
}
```

6.1.4 Tracking

The tracking module starts with initializing all variables needed for tracking and then starts a one-shot timer with a 60 second interval to allow the PIR sensor to warm-up. After this interval the detection module is operational.

The tracking module has to implement a way to keep track of the state of the tracking as mentioned in Section 5.1.5. A simple variable is used that stores the address of the node from which the last message was received. For starting to track a new object, this variable gets a value that cannot represent a node address (for example -1). This variable will also be used to determine the direction of movement when tracking a target.

Chapter 7

Experiments

This chapter describes the experiments and their results conducted with the Mica2 nodes and the infrared sensor. These include experiments done with the entire system as well as experiments done for preparation or (components) testing.

7.1 Gathering sensor data

As part of the analysis of the data obtained from the PIR sensor, data has to be gathered and stored. Both the data from a single sensor as well as data from multiple sensors have to be obtained. The experiments and results are described in the following sections.

7.1.1 Single sensor

As part of the development of an application that can detect objects (persons) with a PIR sensor, the data from the PIR sensor has to be stored and analyzed. This test shows how the data of the PIR sensor look. To gather data from the PIR sensor, a simple nesC program is used to send the sensor data directly to the PC using a programmer and a serial cable. The sensor data is stored on the PC. Later, the data can be plotted using, for example, GNUPlot. Figure 7.1 shows a picture of the programmer and the PIR sensor attached to a post. Plots of sensor data have already been shown in this report, see Figure 3.2 and Figure 5.2.

7.1.2 Multiple sensors

As already indicated the entire system uses multiple sensors. A test setup is used to show the relation between multiple sensors. This setup consists of four sensor nodes that form a straight line. As in the single sensor setup the data from the nodes is stored on a PC. To make the relations between the sensors visible all the nodes have to start sensing at the same time. This is

Figure 7.1: PIR sensor and programmer connected to a post.

done by sending a start message to all the nodes at the same time.

It is not possible to send all samples from four nodes to a single base node. Firstly, the base station cannot process all the messages at the same time and secondly, the individual nodes cannot send 16 individual samples after each other. One solution to this problem is storing the samples in the local measurement flash memory of the node and read back the samples of each node individually. The other solution is to store multiple samples in one message to reduce the number of messages to the base station. The first solution requires the use of the measurement flash memory, which draws a lot of power for writing. This power drain affects the PIR sensor beyond acceptable boundaries (the resulting noise is usually larger than the actual motion detection). The second solution still requires the base node to receive too many messages, especially when the network is extended to more nodes. Furthermore, multiple values in one message require more processing time.

To be able to store the values from multiple sensors, a combination of the afore mentioned solutions has been designed. Each node has an associated node, called a "buddy", which receives the message (containing multiple sensor values) from one node and stores them in its measurement flash memory. The advantage of this combination is that each buddy only needs to receive the messages from one node and it can store the values safely in its measurement flash memory because it does not use a sensor. A disadvantage is that for every sensor you need two nodes, one to connect the sensor to and one buddy. The sensor values are read from all the buddies sequentially.

Figure 7.2 shows the sensor data from four sensors. It can clearly be seen that the object is detected by the sensors with a (more or less) equal interval. Only the time between the detection by node 3 and node 4 is different. This can be the result of a slightly different sensor orientation for sensor 4.

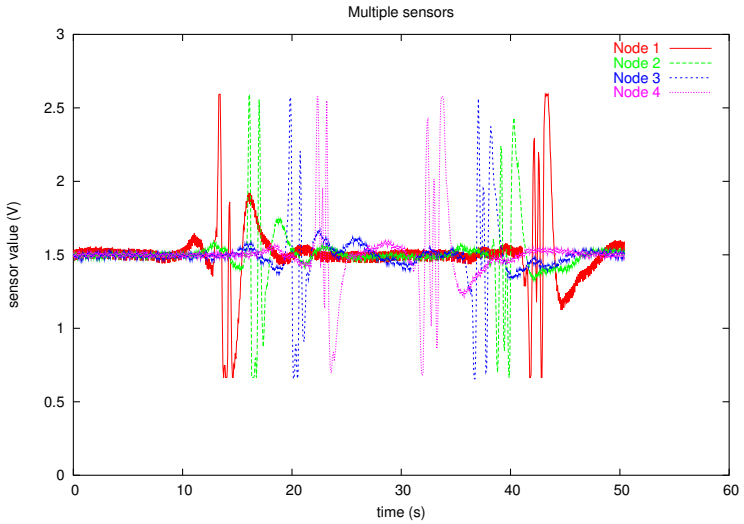

Figure 7.2: Sensor values for multiple sensors detecting the same object.

7.2 Tracking

The experiment with the entire system tests the tracking properties of the system. The setup consists of four nodes in a line topology placed in a hallway. To gather the data, a base node is used which receives all the messages in the network. These messages are stored as a line of byte values in a file to prevent losing messages by processing. These byte values are later converted back to tracking messages. The most interesting values in the tracking messages are the direction and, if the direction is confirmed, the speed and the target id.

The experiment consists of 22 times passing the four sensors with a regular walking speed. After collecting all the messages, the messages with the same target (id) are grouped. For each target it is determined if the direction indication at the first node was correct, and the average speed of the target is calculated. The average speed is the average of the speeds determined by three of the four nodes, the first node cannot determine the speed of an object. The results are shown in Table 7.1.

First direction correct	**Average speed (m/s)**	**Average speed (km/h)**
yes	1,77	6,36
no	1,40	5,04
yes	1,62	5,83
no	1,43	5,15
yes	1,74	6,27
no	1,63	5,88
no	1,31	4,72
yes	2,42	8,72
no	1,67	6,01
yes	6,29	22,63
yes	1,63	5,86
no	1,39	5,01
yes	1,67	6,01
no	1,40	5,03
no	1,07	3,84
yes	1,20	4,30
no	1,11	4,01
no	1,39	4,99
yes	1,60	5,74
no	1,40	5,03
yes	1,61	5,81
no	1,35	4,87
Average (22)	1,73	6,23
Average (20)	1,47	5,29

Table 7.1: Tracking results.

From the results in Table 7.1, it can be seen that the first direction indication was correct in only 50% of the cases. This is less than expected. From all the messages (not shown in this report) it can be seen that if the indication of the first sensor is incorrect, it is mostly mistaking movement to the left with movement to the right.

The direction of movement of an object can always be determined with a 100% accuracy using multiple sensors. Using the relative position information from a node's neighbors and using the information received about a target, the direction of movement can be confirmed starting from the second sensor, see Figure 5.3. Also in Figure 7.2 can be seen that the sensors are activated one after another. Combined with the assumption (see Section 4.3 that each node knows the relative position (left or right) of its neighbors, the direction can be determined by the order in which an object is detected. From Figure 7.2 it can be seen that node 1 detected the object before node 2. Presuming node 1 is on the right of node 2, than the object moves from right to left.

This experiment also show that the speed calculated at the third node is slightly higher than that of the other nodes, which means that the movement is detected relatively earlier at the third node. This can be caused by the sensor orientation. The last two rows of Table 7.1 show the overall average speeds. The final row shows the overall average speed if the two measurements with wrongly calculated speeds (the rows shown in red) are left out.

While tracking a target, the speed is calculated relatively accurately. However, the speed calculation is badly influenced by sensors with a slightly different orientation. The high percentage of wrongly indicated directions at the first node can have different causes, for example the environment in which the tests are executed or the properties of the sensor.

Chapter 8

Conclusions and Recommendations

8.1 Conclusions

The objective of this project is to design and implement an object tracking system using a wireless sensor network. As described in this report and shown in Section 7.2 a system was developed that can track objects (persons) and calculate the speed of the tracked object.

The two main activities done in this project are: connecting a sensor to a Mica2 node and gathering data from the sensor, and designing and implementing the coordination and communication for object tracking.

The PIR sensor used in the system proves to be able to provide a good detection of movement of humans, as was expected. One of the main advantages of this sensor is its analog output. A disadvantage of this sensor is the long warm-up time. Using the built-in ADC of the ATmega128L gives the possibility to connect the output of the PIR sensor directly to the Mica2 node. The improvements over the "Compound Security Demonstrator" relating to the (sensor) hardware are:

- (much) smaller sensor,
- sensor can draw power from the node's batteries and does not require a separate power source,
- the response time (time between detection of objects) is smaller,
- the analog output of the sensor can be used to obtain directional information.

With the algorithm implemented in the system, the sensor network is capable of tracking a single target without using a base station. The role of

the base station is initially reduced to just receiving information from the sensor network.

With respect to the requirements for an object tracking system, the following can be concluded.

- The system is able to detect humans within the sensing range.
- The system can determine the direction and velocity of the detected object. The sensor network can not directly determine the exact location, because the position information is relative, so no absolute position can be given. The location, however, can easily be determined from the detection information (which node has detected the target last) and the velocity. The size of objects cannot be determined using only a PIR sensor.
- Sensors do share information, both to track a target and to confirm information that cannot be determined by a single sensor with a 100% reliability, i.e. direction.
- The nodes report to a single base node. Receiving messages from the base node is not yet implemented in the final tracking program. However, several programs have been used for testing purposes that can perform bidirectional communication with a base station. To demonstrate the operation of the tracking algorithm, the network was not so large that the functionality for advanced network techniques (like multihop) had to be implemented. The base station can be reached from all nodes.
- The system cannot detect and handle disabled or removed nodes. This can be added to the algorithm, for example, by providing nodes with information about the neighbors of their neighbors. So when a node is disabled it can simply be skipped in the algorithm.
- The system has no self-configuration of the network, so the addition and removal (see above) of nodes cannot be detected. Extensions to the network should be specified explicitly in the nodes. Because the tracking algorithm uses relative information and only communicates with its (two) neighbors, for the algorithm, there is no limit to the number of nodes in the network.
- The program has no special functions to ensure low power operation. As mentioned, the used PIR sensor draws its power from the node batteries, so there are no extra batteries needed, this is a form of power saving. Researchers at TNO and at TU Delft are working on an implementation of t-mac on the Mica2, which can save up to 95% in

power only on communication. An addition to the tracking application could be to simplify detection using the digital output of the PIR sensor and wake up neighboring nodes from a sleep state after detection of an object.

- The response time of the system was required to be $\frac{1}{2} \times \frac{d}{v}$, with $v = 1.4$ m/s. This means that for $d = 3$ m a node should report movement within $\frac{1}{2} \times \frac{3}{1.4} \approx 1$ s. experiments show that the speed can be calculated fairly accurate for a walking person, which means the delay between detection and sending a tracking message $<< 1$ s. Otherwise the speed could not be calculated accurately.

Experiments with the object tracking system show that an object can be tracked through the network and the speed of that object can be calculated. Both target information as well as coordination of reporting to a base node are handled within the sensor network, without the use of a base station. Reviewing the requirements the presented object tracking system has the following improvements over the "Compound Security Demonstrator":

- communication between sensor nodes,
- cooperation between sensor nodes,
- independent operation, no need for a base station,
- scalability, the sensor network can be extended to a large number of nodes.

The presented tracking system provides a good foundation for an object tracking system that uses multiple sensors. Also the assumptions upon which this system is based are realistic and usable in practice. The knowledge gained in using non standard sensors and the basic communication scheme can be used in other applications based on sensor networks as well.

8.2 Recommendations

Although the system presented in this report can be extended with many features and all kinds of functions for networking, there are a few additions which would greatly improve the usability.

Probably the most important one is localization for both the network topology and the distance between sensors. With this addition the sensors can be placed more or less at random, the distance between sensors does not need to be constant, and a node's neighbors do not have to be specified in the software. At the same time, localization is one of the most difficult

challenges in sensor networks.

The second addition, which is relatively easy, is adding the possibility to configure the network and detection parameters from a base station (typically a PC with a user interface). This addition provides an easy way to (re)configure the network, add nodes and experiment with detection parameters.

Some other features that can be added include:

- multiple heterogeneous sensors (sensor fusion) for better detection,
- an orientation sensor (compass) to detect differences in sensor orientation,
- multihop for base to network and network to base communication,
- robust packaging of the node and sensor for outdoor use and demonstration purposes.

8.3 Final remarks

The system created for my Master project provides an operational example of an application of sensor networks. This system will be used in a demonstration for the Ministry of Defense in October 2004. TNO have asked me to work for four more months to improve the object tracking, implementing the second addition mentioned in the recommendations, improving the algorithm and adding more network functionality.

Bibliography

[1] Berkeley university of california. http://www.berkeley.edu.

[2] Crossbow technology inc. http://www.xbow.com.

[3] A. Arora et. al. A line in the sand: A wireless sensor network for target detection, classification, and tracking. Technical report, Ohio State University et. al., 2003.

[4] Holger Karl and Adreas Willig. A short survey of wireless sensor networks. Technical Report TKN-03-018, Technical Uiversity Berlen, October 2003.

[5] mica2 datasheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-0%5_A_MICA2.pdf.

[6] Tinyos project page at sourceforge. http://sourceforge.net/projects/tinyos/.

[7] Tinyos. http://webs.cs.berkeley.edu/tos/.

[8] Tinyos tutorial lesson 1: Getting started with tinyos and nesc. http://www.tinyos.net/tinyos-1.x/dos/tutorial/lesson1.html.

[9] G.J.A. van Dijk, M.G. Maris, and A.J. Schoolderman. Compound security demonstrator. Technical report, TNO-FEL, December 2003. version 1.0.