Kindergarten: A Novel Communication Mechanism for Mobile Context-Aware Applications

Nahuel Lofeudo[1], Andrés Fortier[1,2,3], Gustavo Rossi[1,3], Silvia Gordillo[1,4]
[1] LIFIA. Facultad de Informática. UNLP. La Plata, Argentina
{nlofeudo,andres,gustavo,gordillo}@lifia.info.unlp.edu.ar
[2] DSIC. Universidad Politécnica de Valencia. Valencia, España.
[3] Also CONICET, [4] Also CICPBA

1 Introduction

During the last decade, mobile context-aware (CA) applications have been gaining importance and are slowly weaving into everyday life. In a conscious and seamless fashion, we start to get used to incorporate new technologies, appliances and applications that where almost unthinkable ten years ago. The penetration of this kind of applications in the society is mainly due to improvements in hardware and communication areas.

However we are still far from having solved all these issues in these areas. Even tough there are different hardware devices and protocols for wireless communication and position estimation, we still need one that is small enough to be integrated into a wide range of appliances, not only PDAs and Smartphones, but also in active badges or key rings. This device must also have very low power consumption, so that the mobile appliance's battery life is not significantly shortened. Most current wireless communication standards have successfully accomplished the size constraint, being effectively embedded in mobile devices. However, low power consumption is still an unresolved issue which leads us to the necessity of conceiving a new kind of hardware.

During our research we tested existing technologies for suitability in our project, including Wi-Fi (O'Hara & Petrick, 1999) and Bluetooth (Morrow, 2002) which, despite their presence in nearly all current PDAs and Smartphones, we found to be too power-hungry. We also tested GPS (Hofmann-Wellenhof, Lichtenegger & Collins, 2004), whose location sensing capabilities only work outdoors making it unusable inside buildings. Unfortunately we arrived to the conclusion that there is no standard hardware device that can be added to any mobile system and that can be used for both communication and position sensing in a practical manner. For this reason we decided to build our own hardware and software platform to communicate and position mobile devices in an efficient way. In this chapter we will describe the design and implementation of our software/hardware combination, which is designed to provide a balance between network bandwidth, power consumption and roaming capabilities. To complement this work, at the end of the chapter we present an example showing how the hardware is combined with our sensing layer to develop context-aware applications. In particular we will show how to use Kindergarten to provide location-based services (Rao, Minakakis, 2003).

2 Previous Work

In order to understand our needs for new hardware and sensing architectures, we have to consider previous approaches and analyze their weaknesses and strengths. In the next subsections we will discuss some of the most pervasive communication and location mechanisms available today. In particular we will describe Infrared, Bluetooth, Wi-Fi and GPS technologies and examine their characteristics from the point of view of a CA application.

Infrared Port

Maybe the most basic device at the hardware level is the infrared (IR) port, which is primarily considered as a communications interface for small devices such as PDAs or cell phones (Knutson & Brown, 2004). In its most pure form, as defined by the IrDA association (IrDA, n.d.), an infrared port sends and receives data between two devices coding it as a stream of infrared pulses. Each device has a small infrared emitter (an infrared LED) to send the light pulses to the other device and an infrared detector to receive the ones sent to it.

This mechanism leaves to software stacks the responsibility of assembling and disassembling the higher-level data structures to allow applications in both sides of the infrared link to exchange data in a reliable and structured way.

This lack of sophistication at the lowest levels of the communications mechanism can be an advantage to adapt this port to serve other functions. In particular, we can make it act not as a communications mechanism but as a location sensing device. To do this, an infrared emitter is configured to continuously send a fixed stream of infrared data, such as a numeric identifier or a position identifier. These devices could be attached to specific objects or locations in the environment to broadcast their data to any infrared-receiving handheld device that passes by, effectively acting as a beacon in its vicinity. When a user carrying one of these infrared-receiving devices (which could be a Smartphone, a PDA or a specifically designed device) approaches the beacon, it will receive the beacon's signal. Then the software running in the device will be able to decode the information present in the beacon and infer its position.

The approach of using infrared light to tag a position in space has all the intrinsic advantages and disadvantages of this transmission medium. An important advantage is its high spatial accuracy: since the infrared beam can be focused to cover a specific area, only devices within this area will be able to receive the IR signal and thus interpret the data it carries. As a downside, the only event that this kind of position sensing mechanism is capable of detecting is the actual approach of the handheld (receiver) device to the beacon emitter. However a second event, which can be related to the user taking away the handheld device or turning away from the beacon emitter, can be inferred when the receiver ceases to detect the beacon.

Summing up, a location system as the one described above is a good choice when there is a need to tag a precise location, as it provides an acceptable response time and relatively good spatial resolution. It cannot, however, be used to detect the physical position of the handheld device (and thus the position of its user) in wide areas, due to its generally narrow field of view, or in places where no beacon can be received at all. Finally, other disadvantage of a solution employing infrared beacons is its limited bandwidth when used to transfer data. The

standard IrDA specification reaches only up to 115.2 Kbps, and although a high-speed standard exists for data rates of up to 4 Mbps, it is not supported by all devices.

Bluetooth

Bluetooth is a communications mechanism that, only in recent years, has found its way into the market mainstream. Developed in 1999 as a wireless link between low-power mobile devices, its primary goal was to provide reliable data-transmission services without significantly reducing battery life (Morrow, 2002). Even though the bandwidth achieved by Bluetooth is not high enough to compete with other communication methods standards such as Wi-Fi, it is not a problem for applications when no important amounts of data must be transferred, which is the case in most mobile context-aware systems.

Similarly to infrared technology, Bluetooth can also be used to estimate the device position, based on other Bluetooth devices acting as beacons to mark a specific position. The use of radio signals improves on the use of light by employing a communications medium immune to interference by the mere presence of any objects between transmitter and receiver.

To apply this method a desktop computer or an autonomous transmitter must be set up to constantly broadcast its Bluetooth identification, which would be received by the mobile device. The mobile device would then relate the beacon's ID to its physical position in a similar manner to the one used in the infrared approach. A second approach to infer the user's position is to use a triangulation mechanism. In this procedure the relative signal strength (RSSI) of a received frame is used, which can be roughly correlated to the distance from the mobile device to the device sending it.

As a data communications mechanism, Bluetooth works in a master-slave connection (Tanenbaum, 2002) oriented network. A Bluetooth master device can connect with up to other seven slave devices for data transfer and a slave device can connect to only **one** master device, the only exception being a few very specialized devices that can sustain simultaneous connections to different masters. This makes it impossible for a roaming device to hold a data transfer while moving, because in order for a slave (mobile) device to connect to a different master device, it first has to break all connections to the old Bluetooth master to be able to connect to a new one. This problem cannot be bypassed because the closed nature of the Bluetooth stack makes it impossible to access the hardware directly in a standard and portable manner.

In our tests we found that the process of detecting Bluetooth beacons is slow, taking several seconds to complete this process. Also the most common API exposed to the developer by Bluetooth only allows detecting if a certain beacon is present or not. Some Bluetooth stack vendors (notably IVT Corporation (IVT, n.d.), makers of the BlueSoleil Bluetooth stack) provide ad-hoc API calls to obtain the relative signal strength of a received frame, which can be roughly correlated to the distance from the mobile device to the device sending it. By using this signal strength indicator, we can use triangulation mechanisms to estimate the device's physical distance to fixed beacons, and thus infer the device's physical position. Regrettably, this feature is by no means standard in all implementations of the stack, especially for mobile devices such as PocketPC handhelds.

As a side effect, having the Bluetooth adapter constantly scanning the network for new devices makes it unavailable for data communications and can severely reduce the battery lifetime of the mobile device.

Wi-Fi (IEEE standard 802.11b/g)

Wi-Fi is a communication standard first developed to be used in laptop and portable computers (O'Hara & Petrick, 1999). With data speeds of up to 54Mbps (IEEE 802.11g), nothing in the mobile communications hardware market compares with the transfer speeds achievable with 802.11g, making it an excellent wireless communications tool.

In the same way as Bluetooth, a mechanism based on Wi-Fi can be devised to infer the presence of the device within the coverage area of a certain wireless access point by detecting the beacon frames it emits at regular intervals. In addition to this feature, most wireless adapters have the ability to report the signal strength of the beacons received by the various access points. Since this can be done in a standard and relatively portable way, it allows us to roughly estimate the distance between mobile device and access points, and thus estimate its position.

However, even though it was designed with power-saving features in mind, the energy consumption is still too high to use for more than a few minutes in a highly portable device, such as a Smartphone or a PDA. When active, the battery life of a typical PDA decreases to less than an hour and a half, rendering it unacceptable for real-world context-aware applications, where autonomy of at least eight hours is needed.

GPS. Finally, even though it cannot be used as a communication medium, we must consider the Global Positioning System (GPS) (Hofmann-Wellenhof, Lichtenegger & Collins, 2004), which is the de-facto standard for location sensing mechanisms. Even if it remains the only reliable choice whenever the spatial position of the system is needed, it only works in outdoor environments (Kaplan & Hagerty, 2005) making it unsuitable for our needs. Thus we decided to discard this technology in favor of a more suitable one.

3 Introduction to Kindergarten

From the discussion in the previous section we can infer a very clear set of requirements for the new hardware. In particular we need to:

- Build a location estimation mechanism to provide a means for measuring the physical position of a portable device, and thus the position of the user.
- Build a communication service to provide a reliable message-passing mechanism for data interchange between two different mobile devices.
- Update the physical position estimation at a regular time interval, regardless of the mobile system CPU load and data traffic across the wireless network.
- Complete any data interchange operation in a deterministic time. Also no station should be able to flood the network and prevent other stations to make use of it.

- Use as efficiently as possible the available bandwidth provided by the physical communications medium.
- Make the implementation of the protocol small enough to fit in memory-constrained devices, and simple enough to execute quickly even in low-power CPUs.

Since none of the previously mentioned communications and location estimation mechanisms met all the above requirements, we set out to build a completely new communications architecture exactly tailored to our needs. Consequently, we began to evaluate several hardware alternatives to base our design on.

The hardware platform

The first step in the construction of Kindergarten was to decide the wireless communications mechanism on which all services would depend. This step is not a simple one since there are many wireless communication standards (Cooklev, 2004), each one of them incompatible with all the others, operating at different radio frequencies, and with completely different modulation schemes and link level protocols. As Andrew Tanenbaum wrote: "The good thing about standards is that there are so many to choose from" (Tanenbaum, 2002), and of course, for each standard there are sometimes several different implementations in silicon, by different vendors, with different software interfaces, every one of them incompatible with all the others

It is not our purpose to describe in detail all the available technologies currently being offered in the electronics market. A simple search in any of the major portals on the subject can do it better than this chapter. We will, however, briefly describe the ones we evaluated and the reason for discarding them in favor of the chosen alternative.

The first approach was to use the hardware already present in devices already at our disposal. These devices had Bluetooth and Wi-Fi interfaces and the plan was to use these interfaces bypassing the whole built-in radio stacks and writing a more lightweight software for the same hardware.

It was soon apparent that the problem of lack of freely available documentation for the controller chips for both Bluetooth and Wi-Fi was an insurmountable obstacle. In both cases the manufacturers provided no documentation on the hardware interfaces for the baseband controllers.

The next step was to look for other standards and radio communication devices that, even if they required a separate external controller, would give the advantage of complete flexibility over the implementation. Understandably, not all requirements outlined above had equal weight in the election of the final hardware. We decided to privilege flexibility and simplicity of the solution over more performing but complex alternatives. For this purpose, a relatively new microcontroller (the CC2431) was selected. CC2431 is being manufactured by ChipCon, which has recently been acquired by Texas Instruments (TI, n.d.). This microcontroller integrates a radio transceiver for the IEEE 802.15.4 wireless standard, an 8051 CPU, 128KB of flash memory and 8KB of static RAM in the same chip. On top of that, what really made

the CC2431 stand out is that it also integrates a location estimation unit, acting as a hardware co-processor operating independently of the software. Each time a data packet is received by the CC2431's radio interface, the hardware also records the strength of the radio signal that carried the data packet. This signal strength is appended to the data itself, and made available to the application software.

The location estimation unit takes as its input the signal strength of received data of between three and eight other radio stations and their corresponding coordinates in space, and produces the best estimation of its own position. The processing takes a few microseconds and is done in parallel with the execution of the program by the CPU (Fig. 1).

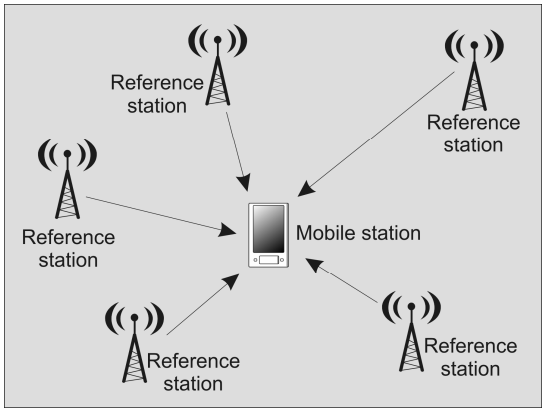

Fig 1. Position estimation in a mobile device

A few words about IEEE 802.15.4

IEEE standard 802.15.4 (IEEE, n.d.) defines the first two layers (Physical and Link) for a low-cost, low-power personal area network. This standard focuses on ubiquitous communication between wireless devices with no extra infrastructure (besides the stations themselves) for several physical layers in the 870 MHz, 900 MHz and 2400 MHz bands, providing data transfer speeds of 20, 40, 100 and 250 Kilobits per second. For low-power embedded devices, the standard provides a tradeoff between speed or coverage range and power consumption to lower even more the needs of the device. Special care has been put forth to ensure the lowest possible manufacturing costs for the controllers, and technological simplicity while maintaining flexibility in the design.

Other important features of IEEE 802.15.4 include real-time suitability by reservation of guaranteed time slots, collision avoidance and integrated support for secure communications. Devices also include power management functions such as link quality and energy detection.

IEEE 802.15.4 is mostly known as the two lowest layers for the ZigBee protocol standard, which aims to create networks of low-power connected devices mainly for home automation and remote sensing (ZigBee, n.d.). An interesting ZigBee feature is the possibility to create "mesh" networks, where all nodes in the network are equally responsible for routing data to

and from any other node. Other operating modes of a ZigBee network are a tree and star-shaped topologies, with one or more "router" nodes coordinating the data flow between "leaf" or low-power, low-performance nodes.

Even though ZigBee could seem as a good starting point for development, given that it already support low-power wireless networks, ZigBee lacks a deterministic timing behavior needed by base stations when sending beacons to update the location estimation in mobile stations. As ZigBee is an inherently asynchronous network, and at the time when the project began there was no free implementation of the stack that supported mesh topology, it was decided that it was a simpler solution to create a new kind of network specifically for our purposes than to re-implement and adapt ZigBee to suit our needs. In recent days Texas Instruments has released a royalty-free ZigBee stack that can be downloaded from their web site. Even so, in our opinion the complexity of the protocol still justifies the development of our proposed solution.

The bird's eye view of the network

With the election of the CC2431 as the network controller the first required task was to create the overall communications architecture. Since the CC2431 can calculate its position relative to other reference stations, we decided to create two different sets of devices based on this controller. One set of devices are base stations, fixed in the environment, which know their physical position. The other set is composed of mobile stations, attached to the mobile device (PDA or Smartphone), or acting autonomously.

As stated earlier, since the position estimation should be refreshed at regular intervals, it was clear from the beginning that a purely asynchronous network (such as CSMA (Tanenbaum, 2002)) would not work. A mechanism was needed to ensure that every controller attached to a portable device would receive a steady supply of packets from the base stations to refresh its position estimation. For this reason we decided to create a slotted synchronous network, where the base stations would coordinate the network access by the mobile stations and transmit the necessary information for them to be aware of their position. The basic premises of the design were:

- The whole network operates synchronously, with no two stations (regardless of being base stations or mobile stations) transmitting at the same time.
- If the physical space covered by a network exceeds the coverage of at least one radio station, the network can be split in multiple sub-networks. It is important that **any** mobile station in a network can receive the data transmitted by **all** base stations belonging to that network. We will discuss this requirement in more detail in the following pages.
- There has to be at least three and at most eight base stations in a network, since these are the absolute minimum and maximum number of references that the location engine of the CC2431 can handle.

In this network, each mobile station receives the signals sent by all base stations in order to update its calculated position, but only communicates with one base station. Thus it acts as a Wi-Fi client, associating itself to a base station and routing all its communications through it. The base stations are wired to an Ethernet network which allows them to receive and send data to each other and to desktop computers (Fig. 2).

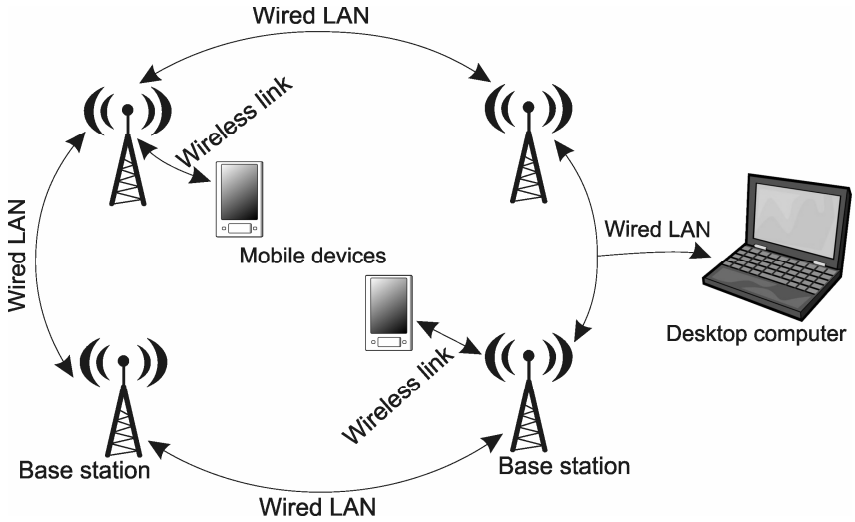

Fig 2. Bird's eye view of Kindergarten

4 Kindergarten's implementation

In this section we will show a detailed description of the network protocol, the principles behind its operation and the different types of frames that stations exchange. We will also address the issue of communication between the CC2431 acting as a mobile station controller and the PDA or smartphone to which it is attached. At the end of the section we compare our approach with other related technologies.

Operation of the network

The stations exchange data through the use of frames, much like any other network in existence. In our case the frames conform to the IEEE 802.15.4 standard, although we use the "packet type" field in the 802.15.4 header to mark our own set of packet types.

As we stated earlier, the whole network operates synchronously. Every base station has a time slot during which it can use the communications channel. The time slot is itself divided in a number of frame slots of fixed size, large enough to accommodate a frame of up to 64 bytes of payload. We have named the time slot of the base station a ***superframe*** to stress the fact that it encompasses the set of data frames sent and received by a base station.

At the start of a superframe, a base station transmits the data frames for the mobile stations. Then it broadcasts to its mobile stations a special frame named “Order Of Transmission” (OOT) which is simply a list of station addresses, in the order they should transmit their data to the base station. Thus, each station knows exactly when to send its data. Since the frame slots are of a fixed size, there will be no collisions with any other transmission from another station.

Once the time for all the data slots has passed, whether or not the data frames had actually been used, the base station sends a second type of special frame, called “Point Of Entry” (POE). This frame serves two functions: firstly it signals the start of a contention period during which new mobile stations can join the network by sending an association request to the mobile station. Secondly, since the POE is transmitted by every base station in every superframe, we use it for piggybacking the location information for that base station so that all mobile stations in range can update their position.

The number of data frames in each superframe is fixed, to ensure that all base stations start and end their superframe in the same time period. Also every station transmits and receives its superframe in its allocated time slot until all stations have done so, and the cycle begins again in an endless loop. We call this cycle **network heartbeat** (Fig. 3)**.**

At a first glance, Kindergarten may seem similar to Token Ring (Love, Siegel, Wilson, 1998), because both networks give each station (base station, in the case of Kindergarten) a time slot to use the communications channel. In this time they can send and receive data, arranged in a sort of ring. However, the similarity between both systems ends there, since in Kindergarten there is no concept of “Token”, and both the sequence of the base stations and the length of the time slots remains fixed regardless of the amount of data to be sent or received by a station. Also, Kindergarten is much simpler than Token Ring, both in its design and operation, so that it can be implemented in devices with limited capabilities.

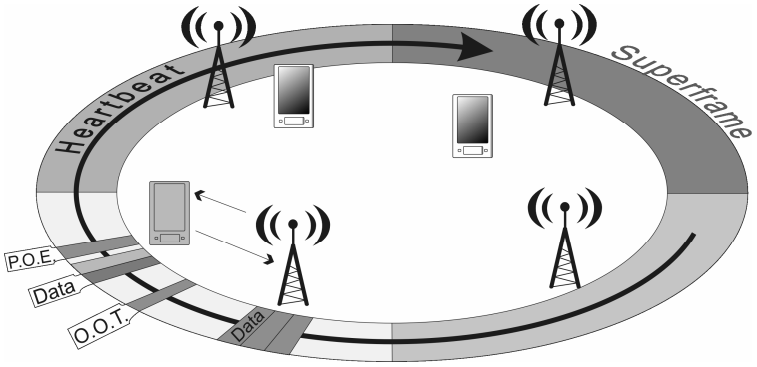

Fig 3. Operation cycle of Kindergarten

This never ending cycle is what gives Kindergarten its name. The analogy can be seen as follows: imagine a group of small children, running around in the playground, shouting at each other while they play. In a sense, wireless stations are like small children: they roam free, shouting (through the wireless channel) when they feel like it, and overlapping (interfering with) each other. Our goal in designing Kindergarten was to create a setting in which

children (mobile stations) are supervised by teachers (base stations) who make them take turns to play.

In case the volume of space to cover is larger than the coverage range of a base station (e.g. when deploying the network in a large hall) we need to scale up the network. Since the location mechanism depends on the mobile station receiving beacons from all base stations in the network, in large spaces it is necessary to split the network in sub-networks operating independently. This is where the synchronous nature of Kindergarten comes into play. The fact that all superframes have the same length in time allows us to create several sub-networks where the base stations are synchronized in a way that ensures they will never interfere with each other.

As an example let us suppose the following setup of two sub-networks with four base stations each (see Figure 4).

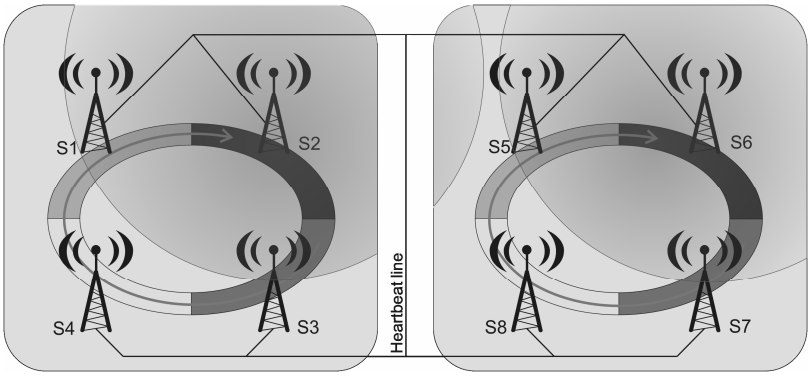

Fig. 4. Non-overlapping simultaneous transmissions

Figure 4 represents the heartbeat with a ring, and the stations with the same position in the superframe (i.e. stations whose superframes are simultaneous) with the same grey level. The coverage area of the stations currently transmitting is shown as a circle around the station.

As this example show, even if both stations S2 and S6 are using the channel in the same instant, since they are outside each other's coverage area their signals will not interfere inside the space occupied by the network (depicted as the grey rectangle). And even if S2 and S5 are inside each other's coverage area they use the channel at different times, thus ensuring that no collision will occur.

Moreover, all base stations are kept synchronized by wiring a second data line through all of them. This data line, the ***heartbeat line***, has as its only purpose to signal the start of a heartbeat by transitioning from a logic HIGH (1) to a logic LOW (0). The rate at which the heartbeat line toggles is determined by the amount of time needed to transmit a single frame times the number of frames per superframe, times the number of superframes in a heartbeat.

Communications architecture

The life cycle of a mobile station begins when it is initialized. At this moment the mobile station waits for a POE (Point Of Entry) frame from any base station and sends an association request to that base station. Then, the mobile station must wait until the next network heartbeat, when the base station to which it sent the request can send a response. If the mobile station receives a rejection frame, it must wait for the POE frame from a different base station and retry the procedure. If the mobile station receives an acceptance response it becomes associated to the base station, routing all data frames through it.

All base stations are wired together to a common communications channel, usually a wired LAN (such as Fast Ethernet), to exchange data frames routed through them. Since all stations in the Kindergarten network operate under very strict timing constraints, it is the responsibility of a base station to enforce the format and timing of its superframe. On the other hand, every mobile station must only transmit when asked to by its base station. Failure of any station to meet these requirements will create interference, forcing retransmission of data frames or mobile stations not receiving OOT or POE frames as long as the interference persists. The stateless nature of Kindergarten makes it possible for the network to recover at the start of the next network heartbeat, when the sequence begins again.

When a mobile station needs to transmit data, it must wait for an OOT with its own address in it. Once the mobile station receives an OOT frame with its own address as part of the frame data, the station must check to which one of the data slots it has been assigned.

A mobile station can only transmit data in a time slot assigned to it by its associated base station, thus once the mobile station has found its time slot it must wait until this time arrives. And since the time slot for a data frame is of fixed length, the amount of data that a station can send is limited to a maximum value. In Kindergarten it is 64 bytes due to limitations in the various buffer sizes in the radio transceiver and the memory space available in the CC2431 microcontroller.

Once the mobile station has sent to the base station a data frame, it must wait for the next heartbeat when its associated base station will send an acknowledge frame (a data frame with a special *type* field). The base station can ask the mobile station for more data in the same superframe, by assigning a new time slot to it, using the same procedure of inserting the mobile station's address to the OOT frame sent in the superframe.

Once a base station has received a data frame, it sends the frame through the wired network to a central desktop computer. This computer acts as a router for all communications, storing the associations between the mobile stations and base stations, and routing the data frames to any mobile station through the appropriate base station. The routing is done using the addressing scheme of the wired interconnection between base stations. In our case, it is done addressing the base stations by their MAC (Ethernet) addresses.

When a mobile station is taken away from the base station through which it sends and receives data, its signal starts to fade. The mobile station, then, requests the central router to switch the data communications to a closer base station through a special data frame addressed to the router. Thus data frames keep being able to travel to and from the mobile station as it moves.

Hardware platform and basic software abstractions

The next step after the design and implementation of a new communications architecture like Kindergarten is the physical construction of the devices, which are the ones to actually carry out the processing. In our case, a large part of the hardware design and implementation work was already done by the makers of CC2431 by creating a chip that could carry out the processing and communication stages in a single package. However, we still have to connect the controller to either the PDA or Smartphone, in the case of a mobile station, or to the wired network, in the case of a base station.

The case of the base station was the quickest to decide in favor of standard LAN technologies, since nowadays almost every building has some kind of Ethernet wiring in place or, in its absence, the cost of running UTP cables through it is sufficiently low as to justify the decision to use it. Moreover, IEEE 802.3 standards (IEEE802.3, n.d.) use only two of the four pairs of wires available in UTP cabling. This leaves the other two (normally unused) pairs free to run the power lines for the base stations and the heartbeat line. We chose to use a relatively new controller to interface the CC2431 and the Ethernet network: the ENC28J60 made by Microchip Corp (MCP, n.d.). This integrated circuit is a complete IEEE 802.3 MAC and PHY controller designed specifically to be used in micro controlled projects through an SPI (Serial Port Interface) interface. SPI is a synchronous serial bus created for data exchange between micro controlled devices (SPI, n.d.). It only needs three data lines (*Clock*, *Master Input Slave Output* and *Master Output Slave Input*) to transfer data at the maximum data rate supported by both endpoints of the bus. This means that there is no need of complex connections between the CC2431 and the ENC28J60. All in all, we use just six data lines between both circuits:

- Clock, Master Input Slave Output and Master Output Slave Input (SPI bus)
- Slave Select line, for the CC2431 to specifically address the ENC28J60
- Interrupt line, for the ENC28J60 to signal an event to the CC2431
- RESET line, to initialize the ENC28J60 once the software in the CC2431 has begun its execution.

If we add two other lines for power supply, this makes a total of eight electrical connections between the CC2431 running the Kindergarten protocol and the ENC28J60 working as Ethernet adapter, enormously simplifying the task of electronics design. As an added value, Microchip Corp. has created a free TCP/IP stack to be used with its ENC28J60. We have used this stack to allow the Kindergarten router (running on a standard PC computer) to exchange data with all base stations through UDP packets.

In the case of mobile devices the choice was not so easy. After surveying the communication mechanisms available in our devices and others available in the market, it was evident that the only common expansion mechanisms for all devices were either wireless (Bluetooth or WiFi), USB (USB, n.d.) or the SD/MMC (SDMMC, n.d.) slot.

We quickly discarded the wireless protocols because that would force the design of the mobile station to include a second radio interface just to communicate with the PDA, raising significantly both the cost and the complexity of the solution.

USB was discarded due to hardware constraints. The basic rate for the USB serial line is 1,5 Mbps, much higher than the data rates achievable with either software or hardware serial ports in the CC2431 and beyond the processing capabilities of the CC2431's 33MHz CPU. We also discarded adding a separate USB controller to the design since it would have made the design more complex and costly.

Consequently we settled in using the SD/MMC slot. On the hardware level, it is a serial communications bus 1 or 4 bit wide, depending on the device's capabilities, operating in a command-response scheme. The same physical bus is used for three separate standards: MMC, SD and SDIO, the first two being purely for data storage, and the third one supporting non-storage devices and I/O operations. We chose SDIO mostly because its basic data rate is 400 Kbps., that can easily be handled by the CC2431's hardware serial ports. The command-response times are also more lax than in the case of USB, allowing sufficiently long times for the CC2431's CPU to do all the necessary processing. In this way, the CC2431 could emulate a SDIO card inserted in the SD/MMC slot of the PocketPC or Smartphone and be detected as such by the operating system.

The implementation of a SDIO card emulator for the CC2431 was based on information publicly available on the Internet: the lower-level SD bus specification is downloadable directly from the SD Card Association (SDCARD, n.d.) and the higher-level functions are described in great detail in the Linux SDIO Stack source code (LNXSD, n.d.).

For the PocketPC / Smartphone side, we developed a SDIO client driver (MSDN, n.d.) to be loaded by the operating system upon the insertion of the card carrying the CC2431. This driver creates a pseudo-file which constitutes the interface to higher-level software functions, and it is through reads and writes to this file that the application software (in our case written in VisualWorks Smalltalk) and the lower-level functions communicate.

At the Smalltalk level we leverage the user of having to think in terms of packets and pseudo files and use a mailbox-based approach to communicate the different hosts in the network. Each host can use three classes to interact with the network:

- A `KindergartenConnection` class, that is used to manage the connection of the host to the network. This class allows the host to get information about the network (e.g. knowing if the host is connected, its address or asking for the available hosts in the network).
- A `KindergartenOutbox` class, that is used to send synchronous or asynchronous messages to other hosts in the network. Asynchronous messages are supported by Kindergarten's hardware, while synchronous messages are emulated. This class takes care of splitting the message in packets of 64 bytes and sending the corresponding commands to the pseudo-file.
- A `KindergartenInbox` class, that is used to queue received messages. The programmer can poll this class for new messages or use it through a publish-subscribe mechanism.

As a result we are able to use high level abstractions to control the proposed hardware, both at the communication and location level.

In the next subsection we will compare Kindergarten with other similar technologies.

Comparison with other protocols and communications hardware

In section 2 we analyzed the advantages and drawbacks of existing hardware technologies for position estimation and data communication. In short, we have found that most "off the shelf" hardware has severe limitations when employed in context-aware applications, the most significant ones being high power needs and low spatial resolution, thus leading us to create Kindergarten.

Technology	Data Bandwidth	Location precision	Power consumption (min/max)
Wi-Fi	11 / 54 Mbps	0.5 Meters	50 uA / 90 mA
Bluetooth	720 Kbps / 2.1 Mbps	20 Meters	15 mA / 60 mA
Infrared	115 kbps / 1.5 Mbps	N/A	0 / 15 mA
GPS	N/A	3 – 5 Meters	90 mA
Kindergarten	250 Kbps	0.5 – 1 Meter (estimated)	0.9 uA / 25 mA

The following table summarizes the main points of section 2:

Table 1: Summary of the different wireless technologies discussed in this chapter

The hardware employed in Kindergarten (the CC2431 microcontroller) has significantly lower power needs, especially in low-power mode, where most of the chip is shut down and only the basic functions remain active, awaiting an external event. Kindergarten is designed for mobile stations to maximize the time where the microcontroller is in low power mode, thus reducing power consumption to a minimum.

In comparison, when used as a location detection mechanism both Bluetooth and Wi-Fi have to scan all available channels for beacons. This forces them to spend a significant time in active mode hopping from channel to channel just to receive said beacons.

In section 3 we introduced Kindergarten as a globally-synchronous network. We have chosen this approach to simplify the protocol and keep its implementation in the CC2431 as lean as possible. By assigning each station (either base or mobile) a fixed slot to send and receive data, we avoid problems and complexity created by collisions in accessing the shared medium.

This approach is similar in spirit to that of Token Ring, where a station can only transmit while it holds a special "token", but is much simpler since there is no token to pass around and is more adequate to the shared medium of wireless networks.

Approaches with "carrier sense multiple access"(CSMA) algorithms like CSMA/CD (Ethernet) or CSMA/CA (Wi-Fi) were discarded because of their nondeterministic nature: each mobile station needs beacons emitted from each of the base stations at a regular rate to update its own estimated position, and CSMA algorithms cannot guarantee their transmission under conditions of heavy network load (Tanenbaum, 2002).

Finally, even though Kindergarten shares similarities with ZigBee, it lacks Kindergarten's deterministic timing and simplicity.

Although no formal Kindergarten network is fully functional at this time, initial measurements of wireless performance on the CC2431 have yielded excellent results.

As for throughput alone, we have achieved speeds of more than 200 frames with a 64 bytes payload per second, with a packet error rate of below 2% over medium distances. With these data it is possible to obtain a rate of 3 superframes per second when each superframe includes 32 slots for data frames, yielding up to 98 Kbps of raw data throughput. Another possible configuration would be to reduce the data slots to 2, which would give us a rate of 20 superframes per second, reaching 41 Kbps of data speed. This allows the implementer of the network to balance the raw data throughput versus the frequency of location updates.

The design and implementation of a new communications infrastructure is not something to be taken lightly. We have struggled to keep the hardware and software design as simple as possible. Early results show promise for this technology although further work is needed to ensure its correct implementation and operation. We have confidence that the low cost (less than five dollars in the case of CC2431) and low complexity of the hardware will encourage others to build their own units and contribute to the development of Kindergarten.

5 Case Study: Using Kindergarten to provide LBSs

In this section we will show how to use Kindergarten combined with our sensing architecture to monitor the user's location and provide Location Based Services (LBSs) (Rao & Minakakis, 2003). For the sake of simplicity we will use a minimal context model that just takes into an account the user's location. Also, since location modeling is not a trivial task and it is out of the scope of this paper to discuss the different approaches (the interested reader can see (Bauer, Becker & Rothermel, 2002; Haibo Hu & Dik-Lun Lee, 2004; Hightower & Borriello, 2001) for further references) we will represent the user's position with a symbolic location model (Leonhardt, 1998). In this model areas are represented as symbols and the relationships between them with graphs and trees (e.g. a tree to establish the containment between areas and a graph to represent adjacency). All these relationships are encapsulated inside a *symbolic map*.

Finally, to provide location-based services we use a set of mappings that associate symbolic areas with a collection of available services. Thus, each time the user's location changes (i.e. he enters a new area) the system must recalculate the available services.

Context Model Overview

To build context aware applications we devised a layered architecture, where the context model resides in a lower layer and the context-dependent behavior sits on an upper layer. Since it is not the purpose of this chapter to discuss our context model or how to provide context-dependent behavior we will just define the basic abstractions used in our context model. The interested reader can obtain more information in (Challiol, Fortier, Gordillo & Rossi, 2007; Fortier, Cañibano, Grigera, Rossi & Gordillo, 2006; Rossi, Gordillo, & Fortier, 2005).

The context model layer contains two main abstractions: the *aware objects* and the *context features*. An aware object is any object whose context is relevant to decide the application's behavior, like a user to provide location based services (LBSs) or a room to provide smart home facilities. Also, since we do not conceive context as a whole, closed object, we split the context into a collection of context features. As a result, in our LBSs example, we represent the user as an aware object and his location as a context feature. Figure 5 shows an object diagram of this situation.

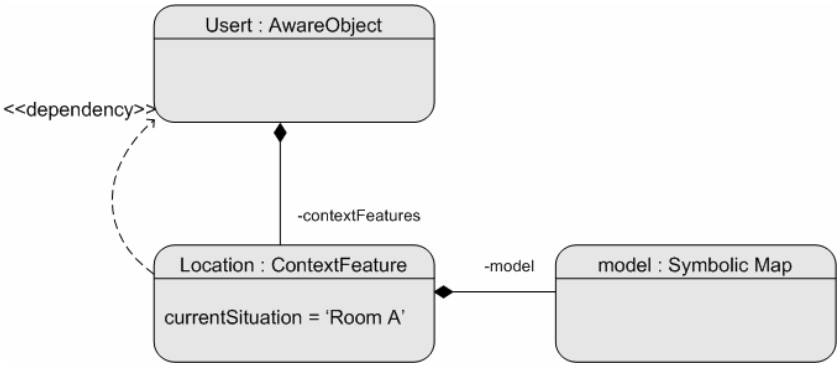

Fig 5. Object diagram of a user with a location context feature

As explained before, we will use a symbolic map of a building, assuming that we already have the required classes for managing symbolic maps (i.e. finding adjacent locations, testing for inclusion and so on). This management can be achieved because the context feature holds a reference to the symbolic map (its model), which gives meaning to the symbol that represents his location.

Sensing Layer Outline

In order to ease the process of gathering sensor data to update the context model we have built a set of abstractions, grouped in a specific layer. As in the previous section, we will give a short description of this layer so that the main concepts are understood. The interested reader can get more detailed information in (Grigera, Fortier, Rossi & Gordillo, 2007).
The purpose of the sensing layer is to help in a set of tasks that are common to most context-aware applications that need to access sensor data. In particular the process of sensing data and feeding the derived information to the context model involves:

- Using a (generally low-level) API. This API is commonly wrapped in an object that represents the physical sensor.
- Configuring sensors acquisition policies. While some may provide a notification mechanism others require constant polling for new data.
- Filtering signals gathered by sensors. In some cases a signal may not fit the required standards (e.g. a measure's error is greater that a certain threshold) while in other

cases the data may be corrupted. Also, like in the case of Kindergarten, we may want to avoid processing multiple consecutive times the same information.

- Converting sensor data to match the context model domain. This step generally involves some sort of abstraction, since most sensors operate with low-level data (bytes, numbers, strings, tuples) and context models composed of semantically-rich objects. It is worth to mention that this transformation can vary in its complexity, ranging from a simple table lookup (e.g. matching a beacon id to a specific area) to complex algorithms (e.g. using machine learning techniques to infer the user's activity).
- Updating the context model. After the signal is converted to the context domain, it must be used to update the corresponding context feature. Once this is done the application can perform its context-dependent behavior (e.g. providing new services).

We next show a class diagram with the main components of this layer (see Figure 6).

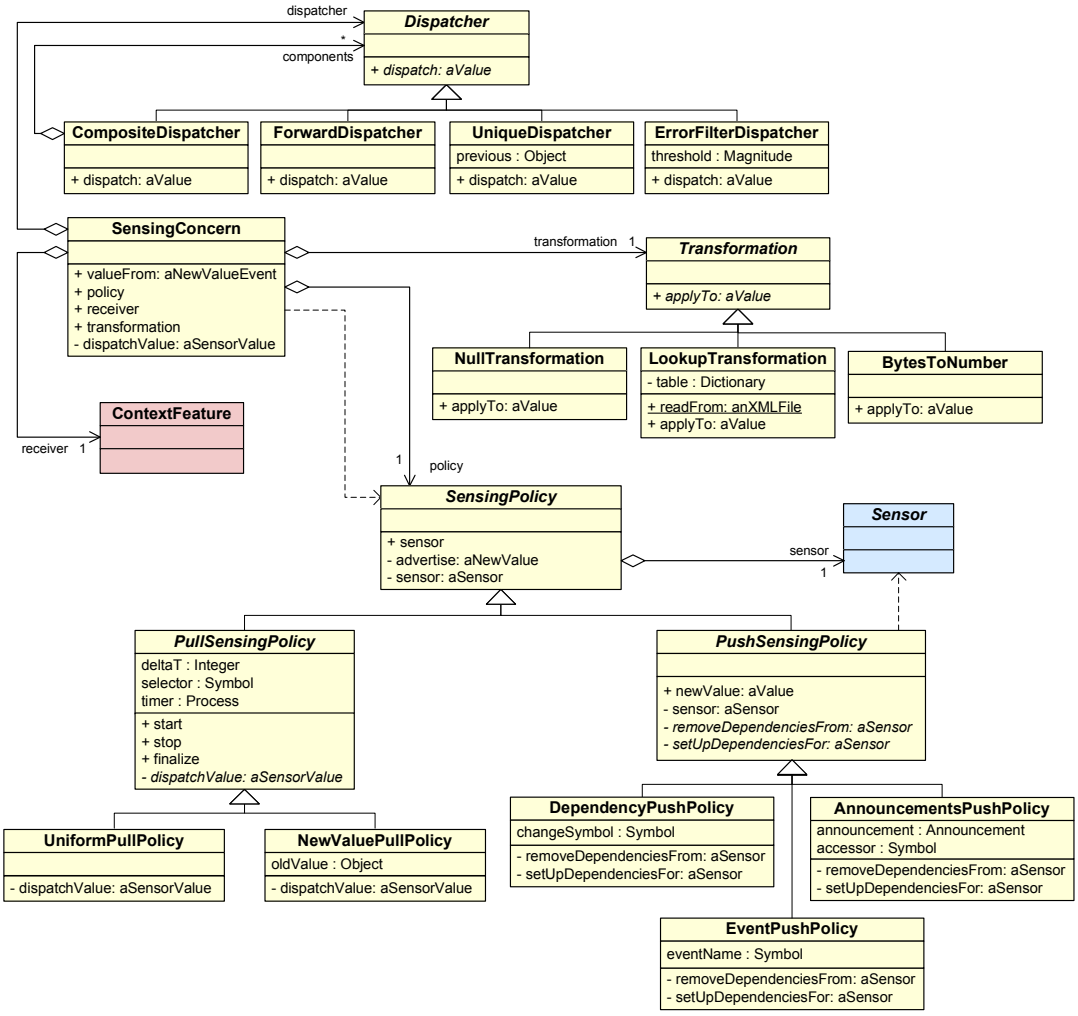

Fig 6. Class diagram of the sensing layer

The entry point of this layer is the `SensingConcern` class, which coordinates the actions of its three collaborators:

- `SensingPolicy`. This class is in charge of abstracting whether the sensor must be polled for data or not, adding a level of indirection between the sensor and the `SensingConcern`. This allows the `SensingConcern` to treat sensors as if all of them used a push policy (i.e. triggering an event each time a new value is sensed).
- `Dispatcher`. Its responsibility is to decide whether a signal should reach the context model or not, acting like a filter. Multiple dispatchers can be combined by means of the `CompositeDispatcher` class, allowing the reuse of already implemented ones.
- `Transformation`. This class is used to map low level data to context information. In case no transformation is needed (e.g. a temperature value is directly used by the context model) a `NullTransformation` is used, according to the Null Object pattern (Woolf, 1997). Other common transformations are already provided by our toolkit, like lookup tables or simple conversions (e.g. interpreting a byte array as an integer or converting a string to a floating point).

After the transformation has been applied, the new object is sent to the context model by the `SensingConcern` class. This step is performed by sending the `#currentSituation:` message to the context feature. Thus, the `SensingConcern` class can be seen as a "big brother", always watching over sensors and deciding when and how a new sensed value will be passed to the context model.

Using Kindergarten to sense the user's position

Having introduced our context model and the sensing layer we can now show how Kindergarten can be used to obtain the user's position while he is moving indoors and translate it to a symbolic location. To obtain the user's coordinates in an indoor space the `KindergartenConnection` class (discussed at the end of section 3) is used. Thus, from the sensing layer point of view, an instance of `KindergartenConnection` acts like a sensor, providing information about the user's location. We next enumerate how the sensing layer is configured to update the context model:

- In Kindergarten we can configure a callback to be executed every time a superframe finishes, retrieving the mobile location according to the chip's location module. This location is retrieved in 2D coordinates, according to the position of the static bases. Thus a simple push policy is created to notify the sensing concern each time the callback is triggered.
- In most of the cases we expect many superframes to be completed without the user changing his position. Thus, to avoid unnecessary recalculations, the location should be only allowed to reach the context model if it is different than the previous one. To

achieve this behavior a `UniqueDispatcher` is used, only propagating the signal if a new position is detected.

- Finally we must map the 2-dimensional coordinates provided by Kindergarten, to the symbolic areas used in the context model. This is achieved by using a custom `Transformation` subclass (`2DToSymbolic`, that uses an RTree to provide fast geometric lookups).

Having set up the context model and the sensing layer to keep it updated, establishing what services should be available for the user at a given time is just a matter of performing a lookup in the set of associations between areas and services.

5 Future Trends

Even though we have a set of prototypes built, we consider our work to be in an early stage. Regarding Kindergarten, we hope to be able to build a prototype network with all its services in place. This network will support mobile devices such as PDAs and Smartphones, and smaller more limited devices packaged in badges or key chains, designed to just report their position to an external system running in a standard computer. This prototype network will let us acquire hard data on the true reliability and throughput of the design, and to make changes in the location estimation mechanism if needed.

At this moment we have yet to tune and test the software that will run on the base stations and in the Kindergarten controller in the mobile station (the previously described CC2431). We are also finishing the device driver that will run in the Windows Mobile powered devices, where the CA application will reside, and allow these applications to interact with other CA applications and the Kindergarten controller.

Our next task will be to create a prototype network using the prototype nodes mentioned above. The aim of this network will be to create a realistic environment upon which we will be able to do real-life measurements of location estimation precision. One of our concerns is the effect that furniture, people and walls will have in the radio signal, and thus the amount of post-processing necessary to sanitize the estimated physical coordinates.

We also wish to test data throughput of the network under real-life load scenarios and to stress test the implementation of the Kindergarten stack.

Regarding the sensing layer we still have a couple of areas that require improvements and different enhancements. Improvements include solving the case of having multiple sensors for the same context feature with different values. An example of this situation could be gathering weather from a web service and from a weather station in a smart home. In this case, the sensing concerns are marked as conflicting and a marshaler decides which of the values will be sent to the context feature (or maybe a new value, derived from the previous ones). In order to choose one value over the other we could use different indicators, like the quality of service, the cost of it or the sampling ratio.

6 Conclusion

In this chapter we have tackled the problem of communication and location estimation for mobile devices, which is a must in any context-aware application. Since our aim is to provide a complete solution for handling external data we also showed how, by means of our sensing layer, Kindergarten can be used to develop very simple location-based services.

As we discussed in the chapter, the implementation of Kindergarten meets the following design requirements:

- Build a location estimation mechanism for a small portable device.
- Build a communication service to communicate such devices.
- Use the transmission medium as efficiently as possible.
- Make the implementation of the protocol small and simple to fit in small CPUs.

Kindergarten is a globally-synchronous network that arranges all stations in an endless sequence called network heartbeat. Each station (be it a base station or a mobile station) has a specific time slot allocated to transmit its information, and all mobile stations refresh their position by triangulating the signal level received from the base stations.

The location estimation mechanism has been designed to be easily integrated to context-aware applications thanks to an event-based mailbox interface. The communications mechanism is fast enough to allow applications to send and receive simple messages, and simple enough to be easily used by other programs. It also allows the user to roam freely while maintaining all communications uninterrupted.

7 References

Bauer, M., Becker, C. & Rothermel, K (2002). *Location Models from the Perspective of Context-Aware Applications and Mobile Ad Hoc Networks*. Personal Ubiquitous Comput. 6, 5-6 (Jan. 2002), 322-328.

Challiol, C., Fortier, A., Gordillo, S. & Rossi, G (2007). A Flexible Architecture for Context-Aware Physical Hypermedia, *dexa*, pp. 590-594, in the 18th International Conference on Database and Expert Systems Applications (DEXA 2007).

Cooklev, T. (2004). *Wireless Communication Standards: A Study of IEEE 802.11, 802.15, and 802.16*. Standards Information Network/ IEEE Press

Dey, A. K. (2000). *Providing architectural support for building context-aware applications*. PhD thesis.

Fahy, P. & Clarke, S. (2004). CASS: Middleware for Mobile, Context-Aware Applications. In *Workshop on Context Awareness at MobiSys 2004*, Boston, MA, USA.

Fortier, A., Cañibano, N., Grigera, J., Rossi, G., & Gordillo, S (2006). An Object-Oriented Approach for Context-Aware Applications. In Proceedings of the 2006 Smalltalk research Conference, Also Springer Verlag, 2006, LNCS.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns*. Addison-Wesley Professional.

Grigera J., Fortier A., Rossi G. & Gordillo S. (2007). *A Modular Architecture for Context Sensing*. In Proceedings of PCAC-07, IEEE Computer Society, 2007, pp.147-152.

Haibo Hu & Dik-Lun Lee (2004). *Semantic location modeling for location navigation in mobile environment* Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on , vol., no., pp. 52-61, 2004

Harter, A., Hopper, A., Steggles, P., Ward, A., & Webster, P. (2002). The anatomy of a context-aware application. *Wireless Networks*, 8(2 - 3):187-197.

Hightower, J & Borriello, G (2001). *Location systems for ubiquitous computing*. IEEE Computer, 34(8):57–66, August 2001.

H Hofmann-Wellenhof, B., Lichtenegger, H., & Collins, J. (2004). *Global Positioning System: Theory and Practice*. Springer.

IEEE (n.d.). *IEEE 802.15.4 task group*, from http://www.ieee802.org/15/pub/TG4.html

IEEE802.3 (n.d.), *IEEE 802.3 (Ethernet) Standard*, from http://standards.ieee.org/getieee802/

IrDA (n.d.), *Infrared Data Association*, from http://irda.org/

IVT (n.d.), *IVT Corporation*, from http://www.ivtcorporation.com

Knutson, C. D. & Brown, J. M. (2004). *IrDA Principles and Protocols: The IrDA Library, Vol. 1*. MCL Press.

Leonhardt, U. (1998). *Supporting Location-Awareness in Open Distributed Systems*. PhD thesis, University of London.

LNXSD, (n.d.). *SDIO Linux Stack*, from http://sourceforge.net/projects/sdio-linux/

Love, R. D., Siegel, M., Wilson, K. T. (1998). *Understanding Token Ring Protocols and Standards*. Artech House Publishers

MCP, (n.d.). *ENC28J60 product page*, from http://www.microchip.com/stellent/idcplg?

IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en022889

Morrow, R (2002). *Bluetooth: Operation and Use*. McGraw-Hill.

MSDN, (n.d.). Microsoft Developers Network, Secure Digital Card Drivers, from http://msdn2.microsoft.com/en-us/library/ms923739.aspx

O'Hara, R. & Petrick, A. (2005). *802.11 Wireless Networks: The Definitive Guide, Second Edition*. IEEE Computer Society.

Rao, B. & Minakakis, L. (2003). *Evolution of mobile location-based services*. Communications of the. ACM 46, 12 (Dec. 2003), 61-65.

Rossi, G., Gordillo, S. & Fortier, A. (2005). Seamless Engineering of Location Aware Services. International Workshop on Context-Aware Mobile Systems, Lecture Notes in Computer Science.

Schmidt, A., Beigl, M., & Hans-W, H. (1999). There is more to context than location. *Computers and Graphics*, 23(6):893-901.

Schmidt, A. & Van Laerhoven, K. (2001). How to build smart appliances? *Personal Communications, IEEE*, 8(4):66-71.

SDCARD, (n.d.). *Secure Digital simplified Physical Layer Specification*, from http://www.sdcard.org/about/memory_card/pls/

SDMMC, (n.d.). *SD Card Association*, from http://www.sdcard.org

SPI, (n.d.). *SPI reference*, from http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Strang, T. & Popien, C. L. (2004). A context modeling survey. In *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*.

Tanenbaum, A. S. (2002). *Computer Networks*. Prentice Hall PTR

TI, (n.d.). *CC2431 product page*, from http://focus.ti.com/docs/prod/folders/print/cc2431.html

USB, (n.d.). *USB Developers page*, from http://www.usb.org/developers

Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using owl. In *Proceedings of the Second IEEE Annual Conference on Perva-*

sive Computing and Communications Workshops (March 14 - 17, 2004). PERCOMW. IEEE Computer Society, Washington, DC, 18.

Woolf, B. (1997). Null object. *Pattern languages of program design 3*, pages 5–18.

ZigBee (n.d.). *Zigbee Alliance*, from http://www.zigbee.org