

# **PLAS.MA**

*Person Loss Avoidance System for Mobile Applications*

Humboldt University Berlin

team members:

Steffen Buhle

Mario Hensel

Dan Kreiser

Johannes Zapotoczky

mentor:

Nikola Milanovic

## 1. Abstract

The PLAS.MA project (**P**erson **L**oss **A**voidance **S**ystem for **M**obile **A**pplications) targets groups of people whose members want to move independently in a certain environment without losing each other. The idea was inspired by the fact that the number of lost or abducted children hasn't decreased noticeably in the last decade.

In May 31, 2003, the Office of Children's Issues was aware of 904 open abduction cases in the United States of America and 78 cases in Germany. There are other statistics (e.g., The National Incidence Studies), pointing to even larger number of abductions. There are no available statistics about the more benign cases of children simply getting lost, since it happens almost every day and usually is not reported. But almost all of abductions/losses happen at day-time and in public places, because the teachers, the kindergarten staff, boy scout guides, were not able to supervise *all* children in their group *all* the time.

Up to now, there have been some simple solutions for that problem (e.g., colored ascots). These solutions work in areas where all group members are visible all the time (not in buildings, for example). This is why we try to construct a more general and progressive solution, with as less hardware and power requirements as possible. Our approach is to build a light-weight and fault-tolerant ad hoc network, which periodically tests whether all members are in range of the group and in case of a lost alarms the supervising person.

The basic idea of PLAS.MA is to tag each person within a group with a radio frequency module (pMod). These tags form an ad hoc network, supervised by one master node. Thus, children in a group can be tagged, and a teacher can observe their movement using a master node. All nodes report periodically to the observing (master) node, but they do not have to have a direct connection to the master. Depending on configuration, they are allowed to go several hops away, and then to report using other nodes (ad hoc communication). The number of allowed hops and the signal strength define the allowed network diameter. If the observing node does not receive a signal from one or more nodes for a certain period of time, an alarm is activated, and the supervisor is instantly aware of missing nodes, as well as of their last known positions and/or recent routes.

The communication protocol between nodes is efficient and simple enough to be implemented in hardware or software, without too complex calculations, saving time and energy. We can guarantee a time interval in which all members of the network will be detected (either as present or absent). All members work as relays to solve the problems of mobility. Only one frequency is used for communication, making this network more resilient to interferences.

PLAS.MA is an easy-to-use system capable of preventing the loss of a person from the mobile group either by accident or through abduction. It is convenient to use with children, especially in places like entertainment parks, museums, etc. However, it is as well configurable for different contexts, and therefore not restricted to kindergarten or school scenarios.

## 2. System overview

### 2.1. *PLAS.MA – a person loss avoidance system for mobile application*

The main functionality of PLAS.MA is not provided by standard computer hardware, but by a special hardware device (pMod), used to tag group members. The idea is to tag children in a group with pMods and then track their movement using one master node, alarming it when one or more children move out of predefined range.

A pMod consists of a Motorola MSP 430 microcontroller and a 433MHz radio module. PMods communicate wirelessly using 433MHz radio modules. One of the pMods (the master) is connected via serial interface to a mobile device (PDA, cell phone). This external mobile device is used as user interface for the supervising person. The master and clients communicate over secure 433MHz radio connection.

The main software modules are: software for the microcontroller ( $\mu$ CSoft), the network and authentication protocols (netProt, authProt) for wireless communication, PDA software-module (pSoft) and serial communication interface between PDA and microcontroller (serIf).

The job of the  $\mu$ CSoft-module is to control the network communication and (in case of the master) to communicate with the PDA (over serIf) and inform its software about the network status.

The network and authentication protocols are responsible for the stable wireless connection between the pMods. They also assure that no intruder can break into the network (only authenticated clients are allowed to be part of the network).

The pSoft provides a graphical user interface on a PDA or a mobile phone. The GUI provides the following functionality: initialization of a network, managing of clients (including a list with personal information) and alarm the supervisor if a client is absent (combined with the information of the last known contact and a pattern of recent movement). It is also possible to send a broadcast message to all group members.

The serial communication interface (serIf) provides all functionality for the communication between the external device (PDA, mobile phone) and master pMod.

### 2.2. *Surveillance aspects*

PLAS.MA has not been built for malicious and unauthorized surveillance applications. Of course in the standard application (holding a group together) every group member is being supervised. But this is done in a context where every user not only knows about that fact, but also definitely wants the supervision to avoid being lost.

An abuse in the sense of surveillance in a way which is not wanted by the supervised person cannot be generally avoided, but the system is not very useful for such purposes. In a surveillance context, the supervisor would have only little information (no exact position of a tagged person, only the knowledge whether the person is in direct or indirect reach of the radio module). That fact makes our system not attractive for surveillance applications.

Of course could the hardware and software be extended (e.g. with a GPS module and a broader communication protocol) and then been used for surveillance. But this would be quite a lot of work and could be done with any communication system.

So we think that our system is not useful for surveillance applications without major modifications. We guess that because of that fact, it is unlikely that anybody would modify it and try to use it for unauthorized surveillance.

### *2.3. Design methodology*

#### 2.3.1. Modularity

One important goal concerning the top-level system design was modularity. At the very beginning we divided the system into four relatively independent modules and declared an expert for every one.

The four modules of PLAS.MA are: communication module (network and authentication protocols (netProt, authProt)), microcontroller module ( $\mu$ CSoft), serial interface between PDA and microcontroller (serIf), and the PDA software (pSoft).

Modularity was important, because we wanted to have a relative independence between the team members. The modularity allowed us to define certain interfaces between system components and then work independently in conformance to those interfaces.

Within each of the four global system components modularity was also important, but not in such a strict way. It was important that every developer organized the work in his part in a way that allowed early testing of functionality.

#### 2.3.2. Early Testing

Another important goal concerning the design methodology was that system and software components could be tested early. So, any functionality could be tested as soon as possible. The modularity of the design supports this goal of course quite well.

### *2.4. Innovation*

The basic idea of PLAS.MA is to tag each person within a group with a radio frequency module (pMod). These tags form an ad hoc network, supervised by one master node.

The main innovative aspects concern both the idea itself as well as the realization. On the one hand it is the idea of holding together a group of persons in which every one wears a small device, without having extra base stations or using foreign networks. This makes PLAS.MA applicable for many contexts where other solutions do not work or are too expensive to build up. On the other hand it is the communication of the ad hoc network, which ensures that all information is sent to the master device. Every device collects information itself and sends this information to its neighbors, so that the information is eventually received by the master (this happens within a guaranteed time). To our best knowledge there exists no device on the market which provides the functionality of PLAS.MA.

On April 17<sup>th</sup>, 2004 Heise Online [1] announced that the Danish company BlueTags [2] has presented a system with Bluetooth tags (BlueTags) for use in amusement parks, zoos and shopping centers. This system is similar to PLAS.MA in the way that persons are tagged and can be prevented from getting lost. But PLAS.MA is significantly different from BlueTags, which works only in environments which have been prepared with base stations. PLAS.MA provides the functionality of holding groups together in any environment, and needs no preparation at all. This makes PLAS.MA also applicable in situations where no infrastructure is available or even installable (e.g. on excursions or hiking tours).

### 3. Implementation and engineering considerations

#### 3.1. Specification of the system

##### 3.1.1. Overview

The network that is spanned by the PLAS.MA system relies on independent specialized hardware devices (pMods). They are managed by the software ( $\mu$ Csoft) that provides the main functionality by controlling the network communication over a specific network protocol (netProt) with support for authentication (authProt) and processing the status data that is exchanged between the individual pMods. Connection to the PDA is established by the master pMod via a serial communication interface. The PDA is the user interface to the supervising person, enabling it to control and survey the PLAS.MA network.

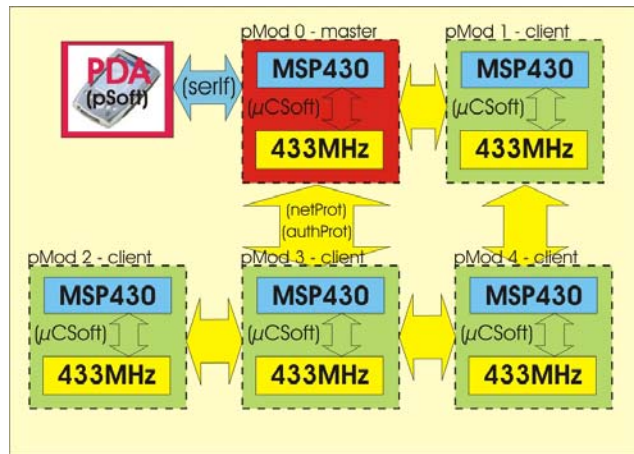


Figure 1: System Overview

##### 3.1.2. Hardware

PLAS.MAs hardware design was to meet special needs regarding the mobility, dependability, connectivity of the individual network units [4].

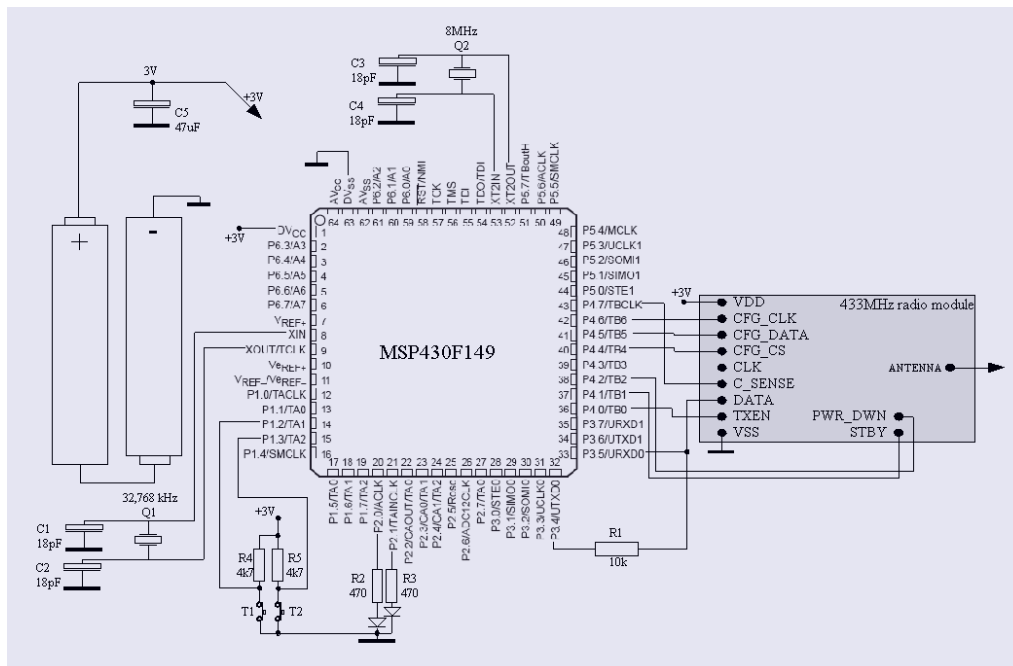


Figure 2: Design of pMod

The core functionality of a single PMod is provided with a microcontroller MSP430F149 from Texas Instruments [5],[6],[7],[11]. We choose this microcontroller because of its well-

balanced characteristic with a fast 16-Bit RISC architecture, integrated universal serial communication interfaces on the one hand and ultra low power consumption with several power saving modes on the other. In addition we equipped it with a fast 8 MHz quartz crystal to fulfill processing speed requirements for communication and processing of network data. It resides on a header board that allows easy access to the programming interface and the input/output facilities. The 433 MHz ISM-Band radio communication is handled by a radio transceiver module with the nRF903 chip from Nordic [8], [9], [10], [12] that provides a well accessible serial communication interface, a configurable output power and a high transmission data rate of 76,8 kBit/s. Combined with two LEDs, buttons and small batteries these components form a single PMod.

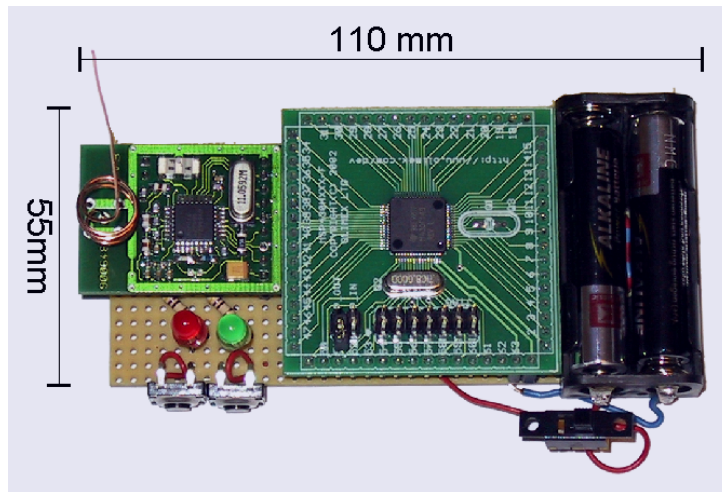


Figure 3: Prototype pMod

The interface to the observing person is established through a PDA, in our case Dell Axim X5, but every other PDA that could be programmed to provide a serial interface via cable connection should be possible. Through the second serial communication interface of the microcontroller that functions as a network interface, the PDA has access to the network.

### 3.1.3. Hardware considerations

The use of the 433MHz radio modules has been carefully considered. We first intended to use the Bluetooth technology, but because PLAS.MA is intended to support long distances between the group members, we decided against Bluetooth. Also, while experimenting with Bluetooth, we noticed considerable problems with detecting new group members, and non-deterministic time variances for establishing connections, which is a serious problem for real-time applications. A next thought was to use wireless LAN instead. However, these modules are quite power-consuming, which is contrary to our idea of having small tags.

We considered using GPS to solve the localization problem for the group members. The main reason why we decided against GPS in this phase is that it is not applicable in closed buildings.

### 3.1.4. Micro controller software ( $\mu$ CSoft)

The heart of PLAS.MA is the software that is running on the MSP430 microcontroller. It is responsible for coordinating the whole network communication between the single Pmods. In case of the master PMod it also establishes the serial connection to the PDA.

Like all software components of PLAS.MA it is built up modularly.  $\mu$ CSoft consists of a core module and modules for wireless communication, serial communication with the PDA and simple human interfaces (buttons, LEDs).

The most important module is the core module, which is responsible for task management and controlling message IO, user interface (buttons), power saving and network data processing. The module for wireless communication provides libraries and functionality to access the radio module hardware. The module for the serial communication is a software module which provides the functionality to establish and keep the serial connection with the PDA. For the access to simple human interfaces  $\mu$ CSoft has software interfaces which make them easy to use.

It should be mentioned that the communication protocols are also modules of  $\mu$ CSoft. They are described very detailed in the next section, so they are not specified in this section and not shown in the following figure.

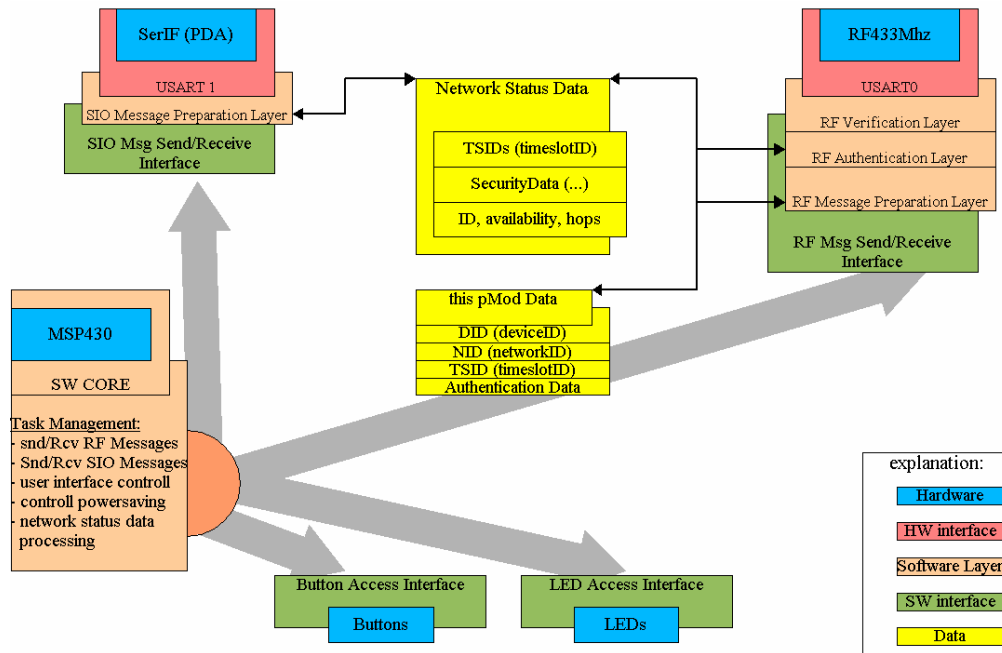


Figure 4: Microcontroller software modules

### 3.1.5. Network protocol (netProt)

The periodical observation of the network members doesn't necessarily require a direct wireless connection between the teacher and the observed child (e.g., Luca can report via Tom on the Figure 5). A child is defined as "out of range" if the group is not able to find it (the child is out of range of every member of the net), or the number of hops (the children that route the communication to the teacher) exceeds a predefined value, or a pMod is removed from the child's body (the device sends a special message). If this happens, an alarm is invoked on the teacher's PDA displaying the name of the lost child and a distance - sorted list of all children which were routing the communication between the teacher and that child

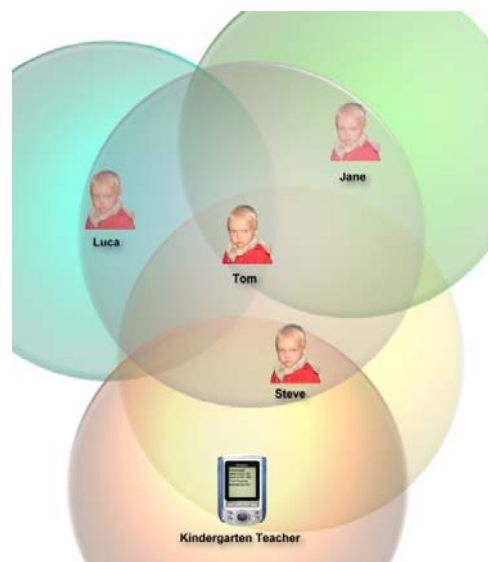


Figure 5: Functioning of the system

in the last searching period, so the teacher is able to judge in which direction the child has gone and knows who to ask for that child (because they were the last who were near to the lost child). Furthermore, the child itself is able to send an alarm message in case of an accident (emergency mode) and the teacher can broadcast predefined messages like “We know want to meet at the bus” (broadcast mode).

The typical steps for communication could be listed as follows:

1. Make the pMod (virtual) a member of a group by sending its MAC-Address to the PDA (called Init0, without that step, the pMod will be unknown at all)
2. Make some pMods out of the available set of pMods physical members of a ad-hoc network by:
  - a. Placing them together
  - b. Run every pMod in the init-mode
  - c. Let the master pMod count all of them (called Init1)
  - d. Let the master pMod transfer the needed communication data to all of them (called Init2)
  - e. Let the master pMod start all synchronously by sending the first session-message
3. Session messages
  - a. Exchange a checklist containing the actual structure of the ad-hoc network (who is near who and who is unreachable)
  - b. Give Errors in case of malfunction
  - c. Broadcast Message codes from the PDA
4. End the communication session by sending the Reset-message from the master pMod.  
(all pMod are once again in the init-mode)

The communication protocol is time-slice based, so devices need to be synchronised throughout the session, because of inconsistencies of their clocks. We use in-communication-synchronisation. Every pMod measures the deviations of incoming messages during the communication (possible because pMod can calculate expected time of message arrival) and calculate the mean value of all deviations. This mean is used to adjust the clock.

**Message Structure:**



*Description:*

*NetIP:* A 16 Bit random-number representing the address of the net. Once this random number is generated, it is used in every message by every member of the net throughout the session, to separate between different groups in the same area.

*ReceiverID:* An 8 Bit number, that specifies the receiver of message. Every pMOD gets a unique Identification-Code (Device Identification - DID) assigned in the initialisation stage. This number is used to address every member of the net separately.

Special DIDs:        0:     The Master-pMOD  
                          255:   All

*SenderID*: A 8 Bit number, that specifies the sender of the message (by its DID).

*Action*: A 4 Bit number representing the content of the message (or the action the message leads to)

Message Types			
Name	Number	Stage	Description
Identify	0	init	pMOD send its MAC-Address
Enum	1	init	First Init-Message: Are you there?
Get_Data	2	Init	Second Init-Message: Here comes the session information
Session_Normal	3	Session (init)	Message is sent in the communication-cycle of the net in a running session
Session_Broadcast	4	Session	Message is sent in the communication-cycle of the net in a running session and contains a additional broadcast message
Session_Error	5	Session	Message is sent in the communication-cycle of the net in a running session and invokes an error output in case of the master as a receiver. The error message is reset to Session_Normal by the master pMOD.
Session_Broadcast_Error	6	Session	The mix of 4 and 5
Session_Reset	7	Session	Resets all pMODs and starts the init-stage
Error_Reset	8	Session	Stops the Error-Mode (just by master pMod)

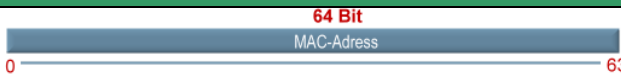
*CS (Checksum)*: 1, if sum of all Bits in message is odd,  
0, otherwise



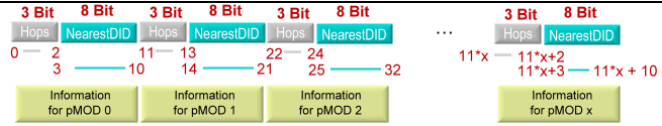
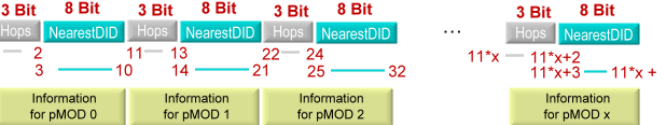
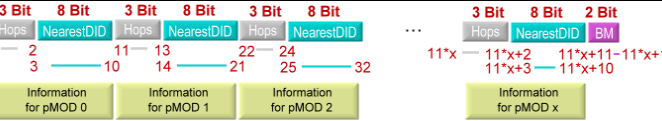
*Authentication sequence*: A 64 Bit number that authenticates message to the receiver. This sequences change every time a message is sent.

*Data*: Additional information (variable in size).

The last block of the message contains additional information depending on the action-number in the message. So it is very variable in size and sometimes even not necessary.

Here is the list of the Action – Data combinations:

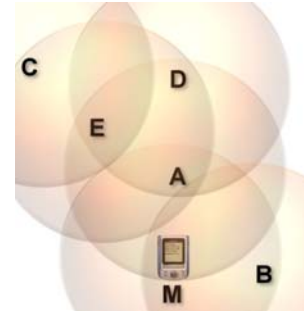
Name	Block-Structure
Identify	 <p>The MAC-Address of the device is transmitted in the data-block</p>

Enum	 <p>The MAC-Address of the device is transmitted in the data-block</p>
Get_Data	 <p>Count: The number of pMODs available in the session (without Master-pMOD). Authentication constants: a, b and m: needed for calculating the Authentication sequence. The same values for every pMOD.</p>
Session_Normal	 <p>The checklist contains the current information of the network. This block is divided into x (=number of pMods in the network, except the master) parts containing:</p> <ol style="list-style-type: none"> <li>1. Hops: The minimum number of intermediate pMODs required to “reach” the certain one. It’s zero, if in the direct neighbourhood.</li> <li>2. NearestDID: The DID of the pMOD that is in range of that certain pMOD and “hear” from it first.</li> </ol>
Session_Error	 <p>Like Session_Normal, but on the PDA an alarm will be invoked.</p>
Session_Broadcast  Session_Broadcast_Error	 <p>Like Session_Normal, but on the PDA will be an alarm invoked and all pMODs will show the message combined with the code “BM” (Broadcast Message). In our first implementation this will turn the LED on that belongs to that code, so the user is able to define the message that is meant by that on its own.</p>

The message is divided into  $n$  parts which means that part  $i$  stands for the net-information of  $pMod_i$ . These parts are divided into *Hops* and *NearestDID* which means that the pMod with the DID *NearestDID* is in the neighbourhood of pMod <sub>$i$</sub>  and *Hops* indicates how many “intermediated transmitting-pMods” are necessary to obtain a message from this member. We always take the lowest number of hops if we have several possibilities, but when we send the number of hops, we increment it. If *Hops* is equal to zero after a communication cycle, then the pMod couldn’t be reached. The message is shown in the following table:

Member 1	Member 2	...	Member $i$	...	Member $a-1$	Member $a$
Hops	DID		Hops	DID	Hops	DID

**Checklist-example:** In order to simplify the example we ignore the NearestDID (just important for PDA to re-construct the path in case of lost), replace the DID-numbers by letters and define 00 as an unreachable member → A1 means: A is “one hop away” from the node which receives this message (the neighbour).



R: this is what the member receives.

S: this is what the member stores.

T: this is what the member is transmitting.

The table shows the time flow of transmitting session messages.

Time (ms)	A			B			C			D			E			MASTER		
	R	S	T	R	S	T	R	S	T	R	S	T	R	S	T	R	S	Status
0		00	A0	A0	A0			00		A0	A0		A0	A0		A0	A0	Not finished
		00	00	00	00			00		00	00		00	00		00	00	
		00	00	00	00			00		00	00		00	00		00	00	
		00	00	00	00			00		00	00		00	00		00	00	
		00	00	00	00			00		00	00		00	00		00	00	
4	A1	00		A0	A1			00		A0			A0			A0		Not finished
	B0	B0			B0			00			00			00			00	
		00			00	00		00			00			00			00	
		00			00	00		00			00			00			00	
		00			00	00		00			00			00			00	
8		00		A0			00	00		A0			00	A0		A0		Not finished
		B0			00		00	00		00			00	00			00	
		00			00		00	C0		00			C0	C0			00	
		00			00		00	00		00			00	00			00	
		00			00		00	00		00			00	00			00	
12	A1	00		A0			00	00		A0	A1		A1	A0		A0		Not finished
		00	B0		00		00	00		00	00		00	00			00	
		00	00		00		00	C0		00	00		00	C0			00	
		D0	D0		00		00	00		00	D0		D0	D0			00	
		00	00		00		00	00		00	00		00	00			00	
16	A1	00		A0			A1	A1		A1	A0		A0	A1		A0		Not finished
		00	B0		00		00	00		00	00		00	00			00	
		C1	C1		00		C1	C1		C1	C1		C0	C1			00	
		D1	D0		00		D1	D1		D1	00		D0	D1			00	
		E0	E0		00		E0	E0		E0	E0		00	E0			00	
20		00	A0	A0	A0		A1		A0	A0		A0	A0		A0	A0	All pMods found!	
		B0	B1	B1	00		00		B1	B1		B1	B1		B1	B1		
		C1	C2	C2	00		C1		C2	C1		C2	C0		C2	C2		
		D0	D1	D1	00		D1		D1	D1		D1	D0		D1	D1		
		E0	E1	E1	00		E0		E1	E0		E1	00		E1	E1		

Messages						
Init Stage						
Name	NetID	ReceiverID	SenderID	Action	Authentication	Data
Init 0	0	0	0	0	None (0)	MAC-Address
	pMod sends its MAC-Address to the Master in the init-stage in order to become a valid member of the master's database. Just the pMods in this database will be used for communication.					
Init 1a	N	S	0	1	None (0)	MAC-Address
	<p><i>N</i> is a random number representing the new valid NetID for the ad-hoc network.</p> <p><i>S</i> is a device-unique 8 Bit number, representing the new DID of the pMod in the communication session.</p> <p>The master counts all available devices by sending this message to all pMods available in database (every known MAC-Address). It also assigns the (physically) available pMods their DID for communication.</p> <p>Important: The devices will recognize <i>their</i> message, by analysing the DATA-part. (Because they don't have a DID to look for in the ReceiverID)</p> <p>Iterate sending message Init1a until all database-known pMods are checked.</p>					
Init 1b	N	0	S	1	None (0)	MAC-Address
	This is the answering-message of the pMod for Init2a to the master. If the master receives this messages, then the pMod is marked for communication (and count). After all devices have been tested, the PDA calculates the 3 constants for authentication: a, b and m.					
Init 2a	N	S	0	2	None (0)	a,b,m,n
	<p>The master sends the constants a, b, m (for authentication) and n (number of pMods in the network without master – important for time-slice calculating) to every single pMod.</p> <p>Iterate sending message Init2a until all physically-known pMods got the constants.</p>					
Init 2b	N	0	S	2	None (0)	a, b, m, n
	Check-message, if everything has been delivered without an error. Otherwise any error would be fatal at this point.					
Session(1)	N	255	0	3	AuthSequence <sub>1</sub>	Checklist <sub>1</sub>
	The master sends the first checklist-message and so starts all timers synchronously on all pMods the time they receive the message.(Assumption: At the beginning all pMods are placed close together)					
Session Stage						
Session(i)	N	255	x	3	AuthSequence <sub>i</sub>	Checklist <sub>i</sub>

	The pMod with the DID x and the timeslot i sends its Checklist in to everybody, who can receive this message.					
Error(i)	N	255	x	5	AuthSequence <sub>i</sub>	Checklist <sub>i</sub>
	Like Session (i), but also invokes an error-message on the PDA. Could be reseted just by PDA itself.					
Error Reset	N	255	0	8	AuthSequence <sub>i</sub>	Checklist <sub>i</sub>
	Like Session(1), but stops error-mode of all pMods.					
Broadcast(i)	N	255	x	4	AuthSequence <sub>i</sub>	Checklist <sub>i</sub> , BM
	Like Session (i), but also sends a Message-Code to every single pMod for Broadcasting.					
ErrorBroadcast (i)	N	255	x	6	AuthSequence <sub>i</sub>	Checklist <sub>i</sub> , BM
	Union of Error(i) and Broadcast(i)					
Reset	N	255	0	7	AuthSequence <sub>i</sub>	-
	Master closes session by resetting all pMods per message to init-stage.					

### 3.1.6. Authentication protocol (authProt)

The PLAS.MA application uses messages to extract information about the actual structure of the ad-hoc network, so it depends a lot on the exchanged messages, which every pMod sends and receives. Every message of the PLAS.MA – device must be verified before progressing in order to reveal messages from an intruder who wants to emulate a certain pMod. For that, we use authentication sequences.

The authentication sequence is a 64 Bit number calculated by every pMod of the net synchronously at every new time-slice (not just on their own time slice) in order to compare the received sequence with their calculated one. If the sequences are different, then the received message is not trustworthy. The formula for the authentication sequence is a small (and fast) one-way function:

$$\text{New\_Sequence} = (a * \text{Old\_Sequence} + b) \text{ mod } m,$$

where a, b and m are 64 Bit constants transmitted in the initialisation steps.

### 3.1.7. Performance requirements

*Time requirements:* The speed for recognising a lost node depends on the number of users in the network and the duration of the used time-slice in communication. We limited the number of net participants to 255, because anything above wouldn't be feasible for one person to observe. In a certain time-slice the following steps occur:

- a. receive the message from the neighbour (worst case): 0.1ms
- b. react on message: 0.3ms
  - i. verify, if trustful  
(compare with actual Authentication sequence)
  - ii. Analyse message and create a new one (worst case)
  - iii. Calculate next Authentication sequence

- c. Send message (worst case) 2.5ms
- d. Time to synchronize 1ms

So we need in total 4 ms for every time-slice (= message) to work properly.

*Space requirements:* Each pMod stores the following data:

- 11 Bit \* n (Checklist)
- +3 \* 64 Bit (Authentication Constants)
- +8 Bit (DID)
- +8 Bit (Number of participants)
- +10Kbyte Code

In total that leads to the need of ≈11Kbyte memory.

### 3.1.8. PDA software (pSoft)

This module is the Graphical User Interface of PLAS.MA .

The application running on the PDA is divided into 4 tabs supporting the user with information concerning the current status of the ad hoc network:

1. Init: The user can select the group which should be observed and starts the session
2. Lost: For each lost member of the group, the user is able just by one click to determine the last persons who were in contact with him
3. List: Shows the current list of all found members in the network.
4. Settings: The user can create/delete network groups and assign members

The application is programmed in C# and runs on every PDA with Microsoft

Windows for PocketPC. The graphical user interface is designed to give as much information as possible on such a small PDA-screen in order to give the user a wide view on the group.

### 3.1.9. Serial interface (serIf)

SerIf, the serial interface between master pMod and PDA provides an easy-to-use software interface for sending and receiving data on the PDA. It allows the PDA software to use all functionality transparently without caring about technical details. All primitives needed for the communication with the master pMod is provided by SerIf.

On the pMod side there is no necessity for an advanced software interface, because sending and receiving on the serial port is natively supported.

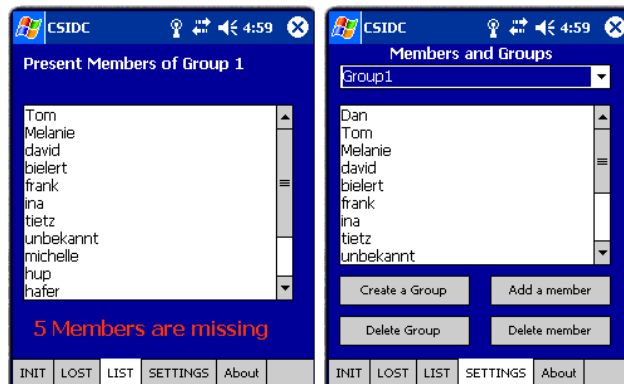


Figure 6: PDA software for group supervision

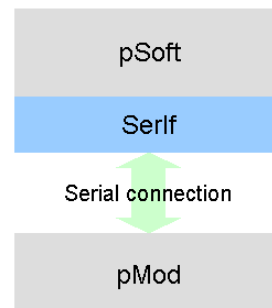


Figure 7: Module for serial interfacing

SerIf consists of three major classes: Serif, Packet and RawPacket. Packet is the data structure used by the PDA software, RawPacket is the data structure for the raw data, which is transferred over the serial connection. The class Serif provides the functionality to connect/disconnect to the

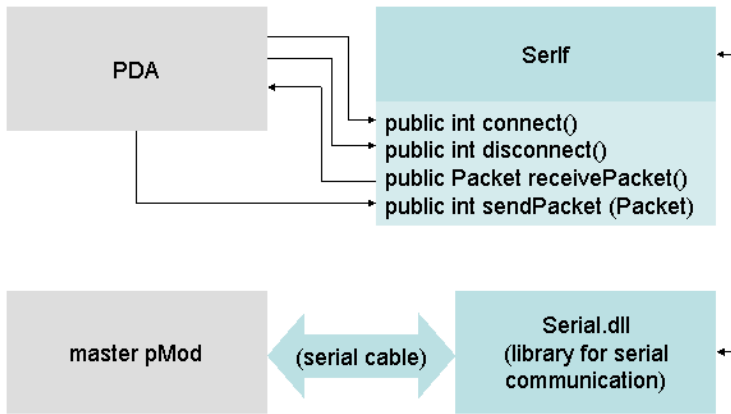


Figure 8: Communication between master pMod and PDA

serial port and receive and send Packets (for use in pSoft). For the low level communication, a library from [3] is used.

### 3.2. Cost

Minimization of costs for a single pMod was an important goal. Of course it is not possible for a prototype to be really cheap, but we managed to reduce the costs for one prototype pMod to less than 75 US-\$. The low price of one pMod allowed us to build five prototype devices. This will enable us to demonstrate the function of the message-hopping between the clients.

<i>List of parts for one module:</i>			
<i>parts</i>	<i>number</i>	<i>unit price</i>	
		<i>€</i>	<i>\$</i>
MSP430-H149 headerboard+µC	1	14,36 €	17,95 \$
FMJA01 ISM-band radio module 433MHz	1	43,50 €	54,38 \$
quartz, 8,0 Mhz (8-HC49U-S)	1	0,49 €	0,61 \$
LED, orange, green	2	0,12 €	0,15 \$
platine 75x100 mm	1	0,87 €	1,09 \$
buttons, different colours	2	0,15 €	0,19 \$
<b>sum</b>		<b>59,76 €</b>	<b>74,71 \$</b>

In an industrial assembly the costs for one pMod can be significantly reduced, because the prototype’s radio module and microprocessor are mounted on header boards which are not necessary in an industrial production process.

### 3.3. Tradeoffs

#### 3.3.1. Limitation of participants

Our system is designed to handle maximum of 255 persons in the network. We limited this number, because the complete observation cycle depends linear on the amount of net-participants and 255 people is such a large group that any extension wouldn’t be useful. Furthermore must kept in mind that just one PDA/master has to “handle” the group, and the lost of maybe 30 out of a group of 255 isn’t feasible, nor does it reflect a realistic situation. In case there are more persons/children that need to be supervised, more that one group should be created.

### 3.3.2. Security aspects

PLAS.MA does not have built-in security in a way that its communication is encrypted. I.e. anyone who has a radio module in the 433MHz band could “listen” to the communication. This is a performance tradeoff. It would take too much computing time to encrypt all data before sending. We thought about that fact a lot and decided that this is tolerable. The “listening” intruder does not get any relevant information out of the communication. There is no position data sent, and an intruder will not know which number represents which person in a group (and even if he knows, that information is not very useful).

Of course we assure that no intruder can send any wrong information in our network. Every sender in our network must authenticate himself – an intruder would not have the correct authentication sequence.

### 3.4. Verification and testing

In a complex system like PLAS.MA, testing plays an important role. Since we used the V-model for our engineering process, we created test cases during every software engineering phase – according to the current phase (e.g. very abstract test cases during system design, very special test cases during fine specification).

Since PLAS.MA has four major modules, and these modules have been developed independently, we did not only extensive testing before starting with the integration, but a whole engineering process on each module. For the network and authentication protocols (which needed a platform to run on) we developed a simulator to be able to test the network communication algorithms in advance, before the hardware was completed.

We also tested the hardware (especially the micro controllers and the radio modules) as soon as it was shipped, and developed some simple tools for this purpose.

After having built and tested all software components on their own, we started the integration process. Of course we did not put all four modules right from the start. Thanks to our module concept, we were able to start the integration process in a parallel way.

In the first integration step, we connected pSoft with serIf (integration step 1.1.) and protocols with µCSoft (step 1.2.) without having any interference.

In the second integration step, we integrated µCSoft with serIf (but only those two, without the other modules (integration step 2)). This integration step was quite laborious – eventually we developed a tool which made it possible to test the serial communication on a PC.

After successful completion of these two integration steps we are currently performing the overall integration and testing.

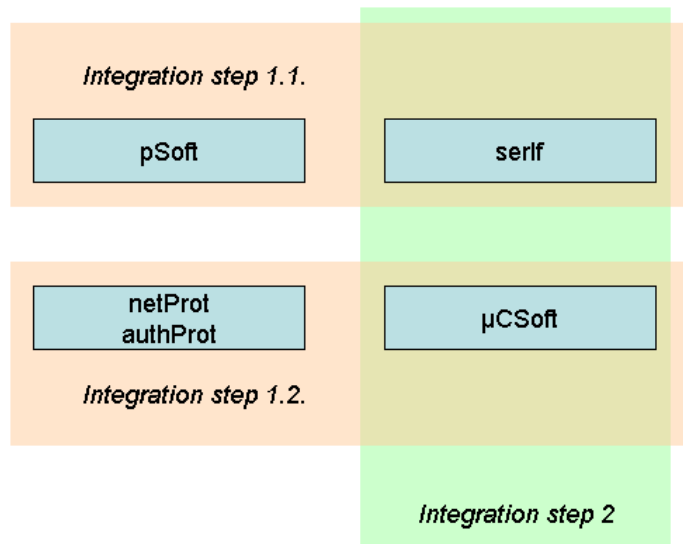


Figure 9: Testing and integration

### *3.5. Description of tools*

For programming the microcontroller we relied on open source MSPGCC development environment, that is specifically suited for the MSP430 series from Texas Instruments, providing extended access to the microcontroller including J-Tag-support for direct programming and debugging. Other software like pSoft, the simulation and testing tools running on either the PDA or PC was developed with MS Visual Studio .NET.

For extended testing of the serial communication between PDA and micro controller, we developed a tool, which runs on the PC and simulates either PDA or micro controller by sending adequate messages. It can also monitor the received data in a very detailed way, so that protocol errors could be detected easily.

To test our network and authentication protocols, we created a simulator. It creates a given number of virtual network members which communicate with each other. The simulator allows parameterization of responsiveness and reaction of every single virtual module according to the real hardware behavior (micro controller and radio module). It is also possible to define the distance between the virtual modules in a two-dimensional manner.

#### 4. Summary

The issue of children safety while playing outside, visiting a museum, or being on the excursion has not been addressed using emerging technologies until now. We tried to use light-weight and low cost computing and communication hardware to provide a solution for tracking children in various environments.

We do so by tagging every child with radio transceiver which periodically reports to the master (observing) node. The tags form an ad hoc network and use each others as relays when transmitting position information. Therefore, this solution is easily deployed without the need for dedicated base stations or other network infrastructure. The master node is instantly alarmed when one or more children move farther than allowed.

We succeeded in building five working prototypes, one of which is connected to the Dell Axim Pocket PC acting as the observing node. We are confident that this solution provides a viable alternative for supervision of large groups of children.

However, we plan further extensions of the model. Our platform can provide data about last known position of a lost child relative to the other members of the group (other children). It cannot provide data about absolute position. Therefore, we plan to integrate GPS receiver on the tag in the future. That should not be difficult, since we already have a microcontroller that processes routing information, and we could utilize it to process GPS information, too. In that case maps of the terrain (amusement park, museum) could be uploaded to the observing PDA, and teacher be warned if any of the group members approaches potentially dangerous place.

Beside GPS integration, we are thinking about adding the possibility of collaborative supervision, where two or more teachers can observe the same group, using two master (PDA) devices.

As with every technology, our solution can be abused. It can be potentially used for unauthorized tracking of people without their consent. However, we have consciously developed a system so that is not feasible for the malicious user to try to rework it in that way, since all messages transmitted through the network give information about relative position inside the group. In case a GPS should be included, the potential of misuse significantly rises. We expect that benefits of the proposed solution will overcome controversial issues of tagging and tracking people. When applied with consent and knowledge of the group participants (children), this product can greatly improve their safety, make the job of teachers and supervisors much easier, make parents more comfortable when sending children on organized trips, and ultimately protect and save children lives.

## 5. References

- [1] Kinder finden mit Bluetooth, News, Heise Online 17.04.2004 - <http://www.heise.de>
- [2] BlueTags - <http://www.bluetags.com>
- [3] OpenNETCF.org - <http://www.opennetcf.org>
- [4] Weber, Matthias: Systematic design of embedded control systems / Matthias Weber. - München [u.a.] : Oldenbourg, 1997
- [5] MSP430x1xx Family, User's Guide, Texas Instruments Literature Number SLAU049, Texas Instruments 2003 – <http://www.ti.com>
- [6] MSP430x13x, MSP430x14x, MSP430x14x1 Mixed Signal Microcontroller, Data Sheet, Texas Instruments Literature Number SLAS272E, Texas Instruments, August 2003 – <http://www.ti.com>
- [7] MSP430 Universal Synchronous Asynchronous Receive/Transmit Communication Interface, Application Report, Texas Instruments Literature Number SLAA049, Texas Instruments, April 1999 – <http://www.ti.com>
- [8] nRF(TM) Radio protocol guidelines, Nordic VLSI ASA December 2002 – <http://www.nvlsi.no>
- [9] nRF903 Single Chip RF Transceiver, Product Specification, Nordic VLSI ASA December 2002 – <http://www.nvlsi.no>
- [10] Antennas for low power Applications, Paper, Kent Smith, RF Monolithics – <http://www.rfm.com/corp/apnotes.html>
- [11] MSP430-Hxxx-E Header Board for MSP430F14x and MSP430F41x Microcontroller, Olimex Ltd. 2002 – <http://www.olimex.com/dev>
- [12] FMJA01 433MHz - Funkmodul, Data Sheet – <http://www.elektronikladen.de/files/fmja01.pdf>