Final Report

Lifetch

Life Saving System

http://www.cs.put.poznan.pl/csidc/2004/

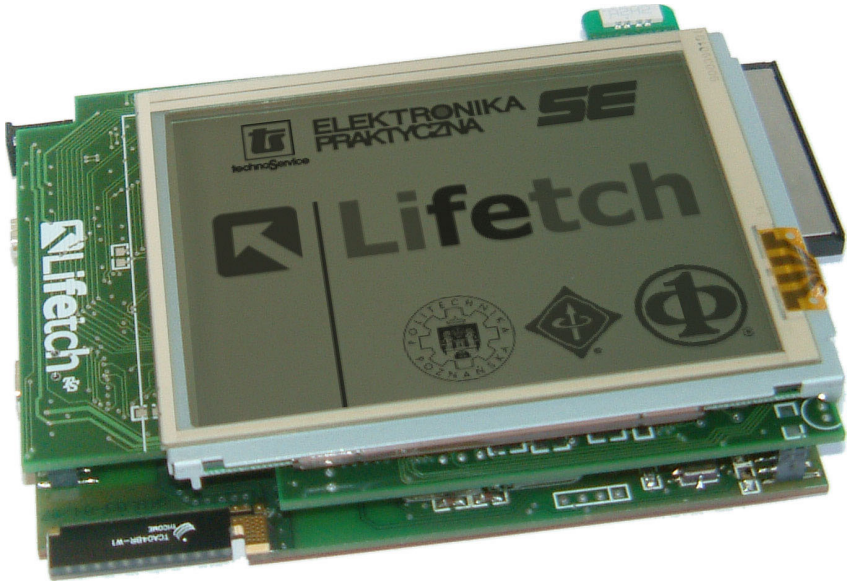
Poznan University of Technology
Poland

Team members:

Wojciech Jaskowski	[wojciech.jaskowski@cs.put.poznan.pl]
Krzysztof Jedrzejek	[goomis@interia.pl]
Bartosz Nyczkowski	[uni@trylion.com]
Stanislaw Skowronek	[sskowron@et.put.poznan.pl]

Mentor:

Jan Kniat, Ph. D.	[jan.kniat@put.poznan.pl]



1. Abstract

Regardless of fast development of technology and human knowledge, year by year statistics of accident casualties in sparsely populated areas often visited by hikers, such as national parks and mountainous terrains, remain high. Existing safety precautions are clearly insufficient. Our research showed that there is currently no system operating over a vast area that improves safety through fast and effective reaction to an emergency situation regardless of the type of the accident and capability of the user to call for help. As of now, lack of feedback from people under threat or those near them makes detecting emergencies virtually impossible.

Numerous consultations, brainstorming sessions and a needs analysis led us to the idea of designing the system solving many of these problems – **Lifetch**. It is based on distributed personal units (we call them **ICU**s – Intelligent Communication Units – pronounced as "I see you") that we designed and built. They combine modules such as: a GPS receiver, a Radio Frequency (RF) transceiver and sensors that measure temperature, ambient light and acceleration. These lightweight units carried by people under protection of the **Lifetch** system communicate with each other over RF and exchange information gathered from the sources mentioned above. The ICUs periodically transmit their data to the Command Center using GSM/GPRS/UMTS (later denoted GPRS as the most practical choice) or, should it fail, the message passing system (ad-hoc network) working on a RF. The Command Center is the heart of our system and maintains its global status. It stores the information acquired from the ICUs in the database and processes it through several subsystems. The goal of these activities is to help the operator ensure the safety of the people protected by the system and automatize certain preventive safety-improving actions.

In order to find out whether our project is viable, we decided on a real-life approach to establish the benefits that would come from introducing the **Lifetch** system. Our needs analysis was based on research of the archival materials of the Polish Mountain Rescue Organization. We selected over 200 mountain accident cases from the last 10 years. Over 80 of these cases included fatalities. The outcome of the analysis showed that in 71% of these cases the effects of accidents could be minimized or even avoided if our system had been deployed.

Improvements raising the safety level apply to the following matters:

- Any person carrying an ICU can **signal an emergency** situation and **request help** even when there is no GSM network coverage.
- Our system **automatically** discovers potentially dangerous situations such as people straying from the safe track and opens more possibilities to react to these situations.
- Implementing intelligent **position prediction** algorithms, the system enables the park rangers to reduce substantially the time of locating a missing person, therefore increasing his/her chances of survival.
- The system gives the possibility to **monitor and analyze the traffic** in national parks or other areas. This can become an important factor while making decisions concerning further safety improvements. It can also be useful for enhancing environment protection.
- The system puts a strong emphasis on **cooperation of a group** of people. Group leader has full information about the status of his/her group, whereas a member can easily locate the group and find the right track even in difficult weather conditions. This can be particularly useful for groups of children on a class trip, tourists hiking with a guide, as well as for a team of rangers performing a rescue operation.

2. System overview

The Lifetch system consists of the following modules interacting with each other: **ICU, Command Center (CC) and optionally a Relay.** It uses the existing commercial **cellular network as a communication medium.**

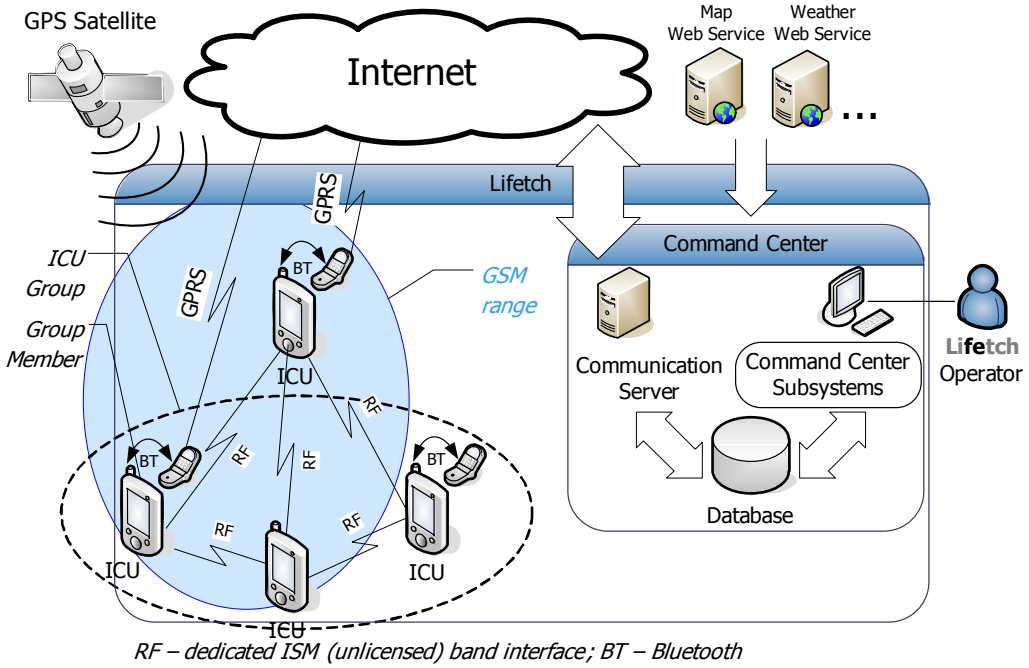


Figure 2.1 Overall system organization

2.1. System hardware elements

ICU – Intelligent Communication Unit. Each person under **Lifetch** protection carries one ICU unit. The ICU contains a GPS receiver, an RF transceiver and other sensors. The ICUs exchange data (messages containing processed information from their sensors) **between themselves and possibly a Relay**. The ICU users may form groups, and ICU is used as a monitoring device for the group leader allowing him to control the status of group members.

ICUs communicate with the Command Center through GPRS capable cellular phones that most of the system users have. The interface between an ICU unit and a phone is a Bluetooth or optionally a USB link.

Relay – has an Internet connection and communicates with the ICUs through RF. Relay serves as an alternative gateway between the distributed RF network and the Command Center.

2.2. Command Center software

Emergency Situation Detection Subsystem – distributed rule-based system detecting potentially dangerous situations based on correlations in data gathered from ICUs.

Position Prediction Subsystem – this is terrain-aware software establishing the probable position of a missing person with significant accuracy, relying on his/her last reported position, his/her latest direction and velocity (calculation based on location history).

Message Distribution Subsystem – broadcasting event messages to all people in a certain area – e.g. about forecasted rapid changes in weather conditions.

On-demand Information Service Subsystem – provides ICU users with requested maps and other information.

Administration Subsystem – allows the operator to perform maintenance tasks.

2.3. Non-functional requirements

The primary goal defined for our system is **reliability** and **stability** of operation. System **scalability** and high degree of integration is required. All wireless communication must be secure against sniffing and spoofing attacks to ensure **privacy** protection. The architecture must allow **integration** with existing systems such as weather services or map providers.

2.4. Innovative concepts

Cellular telecommunication technology is treated as one of the means to solve the safety problem. Undisputedly, many lives have been saved thanks to the services it provides. However, we observed four major shortcomings that limit its application in the field of assuring safety, which do not occur in the **Lifetch** system:

- To call for help, the person in danger must be physically able to do it (conscious) and furthermore must be in the GSM or another cellular network system range.
- Localization techniques based on GSM might not work or would provide low precision in sparsely populated areas. Technically, functionalities of the **Lifetch** system exceed these of enhanced 911 (112 in continental Europe) emergency service.
- An agreement with the GSM network operator is necessary to access location data (if at all possible), which hinders small non-profit organizations.
- On the other hand, satellite systems (like SARSAT) do not provide communication capabilities beyond the basic 'call for help'. Furthermore, the current implementations require the user to manually signal an emergency situation.

The advantage of our project lies also in combining **location, multi-sensor and group cooperation** services in a single device (ICU), which is not possible by means of a standard PDA. Additionally, unlike PDAs available on the market, our units communicate over an **independent radio interface** and have an **open architecture**, allowing them to be extended with, for example, additional sensors. Moreover, through multiprocessor architecture of the ICU, we have achieved a **power saving effect**, providing substantially longer operation time than any PDA. ICUs are interconnected through a **distributed ad-hoc network** and submit data for system-wide analysis to a monitoring system acquiring information also from maps, weather services and other sources. Those are the unique functionalities realized by our system through a specialized hardware unit:

- Our units communicate with the **Command Center** regardless of the condition of the unit owner, transmitting the information about his/her state and location.
- To alleviate the consequences of losing GSM coverage we are using distributed message passing. The ICUs form a **mobile ad-hoc network** (MANET).
- To decide upon signaling the alarm the system uses a rule-based expert system.

2.5. Design methodology

As the design methodology we chose agile methodology based on selected Extreme Programming (XP) practices [5, 6] since it is suitable for explorative projects and small teams. As we had no customer we did not use planning game and other customer related practices. We used UML notation for discussing alternative solutions instead of CRC cards advocated by XP people. To reduce risk of failure we created spike solutions.

As in XP, we applied an incremental approach to the system integration and often refactorized the code, which allowed us to smoothly link all the software and hardware parts of our system together. Furthermore, as we concentrated on reliability and robustness, we benefited a lot from test driven development (test-first coding) and pair programming. At the beginning we performed a needs analysis and generally specified the system requirements. The testing phase was performed both concurrently during implementation, and after integrating software modules of the system. Separate set of tests for the hardware device was also executed.

3. System implementation

The rationale for the system, its design and implementation is detailed in this section.

3.1. User scenario

The user scenario below demonstrates the one of many possible circumstances and the benefit of **Lifetch** for preventing loss of human lives, which is the objective of the system.

On a beautiful Sunday John decided to go hiking with his friend, Mike. At the entrance of the National Park near his town they rented one ICU. Mike also had a cell phone. After a short time they passed a Relay, where the ICU downloaded a general map of the park area. After five hours the ICU started beeping, receiving the message through Mike's phone data transmission line. A warning message arrived informing him of an impending sudden change in weather conditions. He asked through the ICU for the nearest shelter. After receiving a detailed map they headed towards the shelter. Temperature started to drop, a strong wind broke out, and a downpour began. Suddenly, John slipped on a wet rock and tumbled down the steep slope, with the ICU in his backpack. He was knocked unconscious during the fall. Mike yelled for help, also tried to use his cell phone, but there was no response. The situation became helpless, but after a little over twenty minutes Mike heard the sound of a rescue helicopter. They were found and John was taken into hospital. When John regained consciousness they learned that, were it not for the ICU, they would not be found at all. The phone could not function because the base station was damaged during the storm. The ICU detected John's tumbling thanks to the built-in sensors and used radio transceiver to send the message to the Command Center by the chain of ICUs belonging to other hikers. The operator immediately dispatched a rescue team. The whole adventure found its happy end.

3.2. ICU device

3.2.1. Design objectives

To meet the **Lifetch** objectives we formulate the following requirements for ICU:

- the ICU **must** have a radio interface operating in the license-free ISM band,
- the ICU **may** have an interface to a public data network through a cellular phone,
- the ICU **must** be able to perform routing functions between its interfaces,
- the ICU **must** be able to receive messages at any time,
- the ICU **may** perform other functions than safety (e.g. map display, audio playback),
- the ICU **must not** be heavy or in another way interfere with its user's normal activity,
- the ICU **must not** require the user to recharge its batteries often.

Considering these objectives, we have designed architecture of the ICU shown on fig. 3.1.

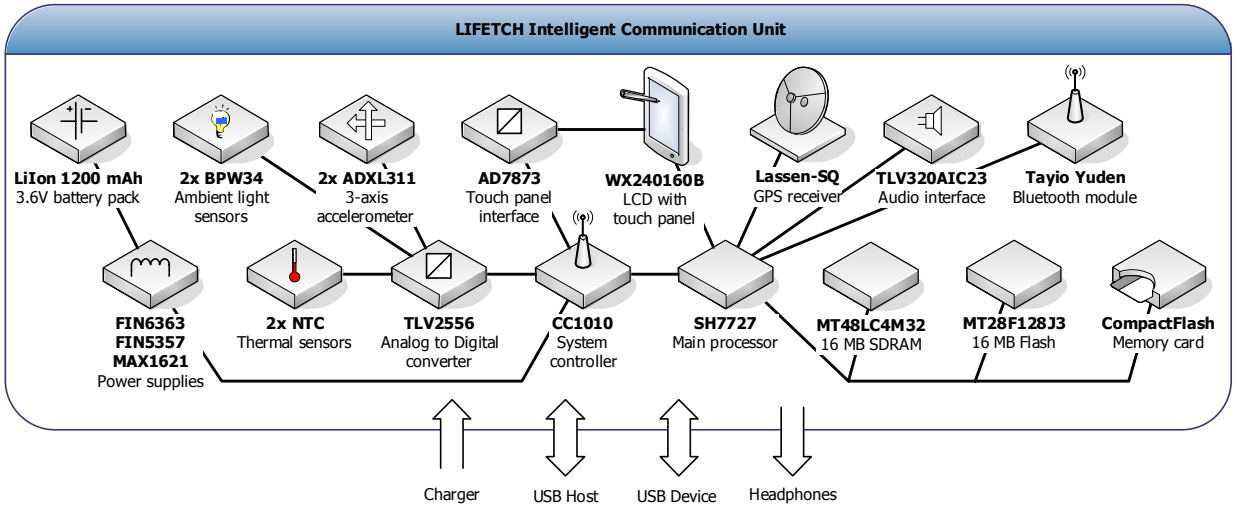


Figure 3.1 Graphical overview of the ICU architecture

3.2.2. The main processor

The ICU performs a set of primary, safety-related functions and also additional, auxiliary functions. The execution of the primary functions must not be disturbed by the other ones. This in turn necessitates the use of a processor that has an MMU (Memory Management Unit), which provides inter-process memory protection.

Thus, a processor that has a true MMU and still is a low-power device had to be found. There were three main candidates: Renesas SuperH, IDT MIPS and a processor implemented in a FPGA. The trade-off between the unsurpassed scale of integration offered by FPGAs and extremely low power consumption of standard devices was made. We decided to use the latter. We chose SuperH, because the IDT chips are targeted towards wired networking. They have many extraneous features that would consume power needlessly in our application.

Because the overall device size is important, it was crucial to integrate as many peripherals as possible into the main processor. The newest member of the SH3-DSP family, SH7727, which is our choice, supports many useful functions: LCD control, audio interface, SDRAM access and others.

3.2.3. Radio interface and system controller

As the ICU has to monitor the radio interface continuously for transmissions, the processor would be kept in a running state most of the time. Taking into account the supply current of the main processor this was unacceptable. Therefore, we decided to use an RF transceiver with an integrated microcontroller. The highest level of integration was provided by ChipCon in their CC1010 device. In order to increase the transmission range we used a Maxim MAX2235 power amplifier. It can output up to 1 W (in Europe the power is limited to 100 mW because of legal restrictions).

A similar issue pertains to sensor readback. For instance, it is necessary to monitor the acceleration sensor continuously, to register peak acceleration. Devoting another processor to this task was not an option, both because of the ICU's power consumption and its size. So the A/D converter and the sensors have been connected to the system controller.

3.2.4. Inter-processor interaction

The two processors are primarily linked by a serial port. But the system controller also manages the power supply of the main processor. If a message arrives and the system controller decides it should be processed, it wakes up the main processor (if it is not already up) and passes the message via the serial port. The main processor can also schedule to be woken up after a specified amount of time or on user request.

The serial port protocol provides guaranteed delivery between the devices. Because there was no latency issue, we chose to implement a ping-pong acknowledgement model (the sender does not transmit any new messages until the last one has been acknowledged by the receiver) instead of the window model.

The inter-processor link layer protocol (we called it IICP – Intra-ICU Communications Protocol) was designed on the basis of the concept of endpoints. An endpoint represents a single logical function. For instance, the system controller contains a measurement endpoint that reports sensor values, and an RF endpoint that performs radio transmission and reception, a user interface endpoint that controls the touch panel and changes the contrast of the LCD. In other words, the endpoint is an abstraction of a functional block.

In the main processor the endpoints are demultiplexed into separate streams. We faced a trade-off between using a demultiplexer that is built into the application and creating a separate process for demultiplexing. We decided that the latter solution is better, although it has more processing overhead, as it precludes the possibility of a single process to block the entire communication protocol.

3.2.5. Board layout

Taking into account the limitations of the size of the device, two stacked printed circuit boards (PCBs) have been designed. This allowed us to parallelize the design process (we prototyped one board while the other has been produced).

The lower PCB is the analog board, centered around the system controller. It carries the RF transceiver, touch-panel controller, sensors, A/D converter and power supplies. The GPS module was placed on this board because of limited space. The analog board can be tested without any external components by connecting it to a PC. This board is a two-sided PCB.

The upper PCB (the digital board) is essentially a single-board computer based on the main processor. It contains 16 MB of flash memory, 16 MB of SDRAM and a CompactFlash card socket. A dual UART is also provided to extend the total number of serial ports to 4. One of these is connected to the GPS module, the second controls the on-board Bluetooth module and the third is connected to the system controller. The digital board is a four-layer PCB.

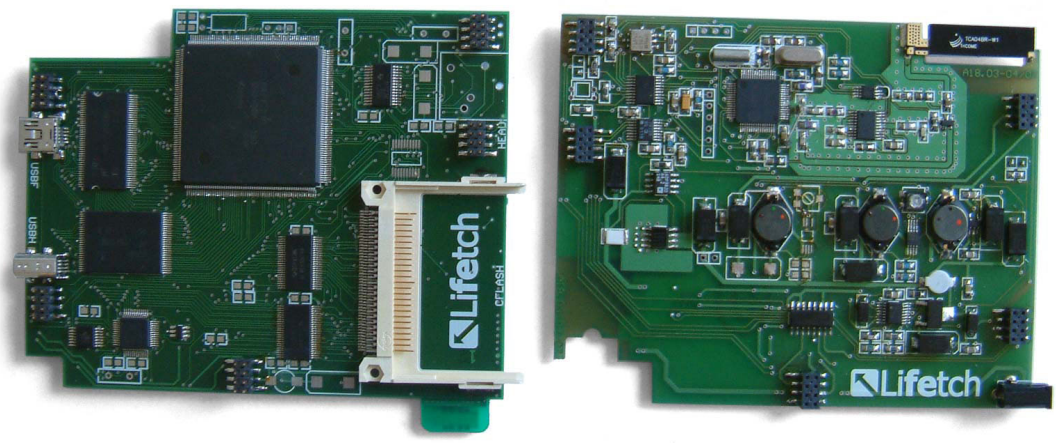


Figure 3.2 Photograph of the digital and analog boards of the ICU in the testing stage

3.2.6. RF design

The RF section of the analog board consists of two main subsystems: the integrated transceiver of the system controller and the power amplifier (MAX2235).

The receiver part of the transceiver operates in the superheterodyne topology, in which the received RF signal is down-converted into Intermediate Frequency (IF) by mixing the signal with a Local Oscillator (LO) output. The outcome IF signal is easier to process. A built-in LNA (Low Noise Amplifier) is integrated into the device. The LO is a typical integer-N PLL.

The same LO is used for FSK (Frequency Shift Keying) transmission. The system controller has an integrated PA (Power Amplifier), however we found its 10 mW output power insufficient. Such a power would impose very high requirements on the ad-hoc network density due to low distances between communicating ICUs. We have attached an external PA. It outputs up to 1W and can be controlled digitally via a Digital to Analog Converter.

With operating frequencies of almost 1 GHz and high generated powers a proper high-frequency layout is crucial. All RF traces are controlled-impedance coplanar waveguides with a solid ground plane. We chose this layout because it is very good for implementing low-noise circuits on two-sided glass-epoxy PCBs, which are relatively thick (1.6 mm) and with microstrip the characteristic impedance of 50 ohms is realized as an over-2-mm-wide track.

After calculating the return losses on the RF path we found that even using a directional coupler, a substantial amount of output RF power will return to the receiver instead of being radiated from the antenna. This reflected power would be enough to disturb the balance of the PLL. Its stability might be affected and transmission spectrum would deteriorate. This is why we placed a PIN-diode transmit/receive switch that isolates the power output from the receiver input. The system controller drives this switch.

3.2.7. User interface hardware

The user interface is provided by a 320x240 backlit LCD with a touch panel. This display may seem big for a simple safety device. Yet, we foresee that the ICU may be used in many more applications. We have found that map readability benefits from the 0.24 mm pixel pitch in comparison to conventional 0.35 mm displays. The display is controlled directly by the main processor. However, because the user initiates interaction with the ICU, the touch panel has to be monitored continuously. It was achieved by connecting it to the system controller.

3.2.8. Power supply

All device elements can be separately powered up and down by the system controller. This imposed certain requirements on the power supply. A switching power supply was used, because it performs well under changing loads. Also, a lithium battery charger has been incorporated into the design to produce a truly integrated device.

The prototype uses a 1200 mAh Li-ion battery from an Ericsson T20 phone. During battery lifetime tests we simulated the nominal ad-hoc network density and the operating time of the prototype was about 36 hours. However, it is conceivable that the production version, using integrated devices with lower power requirements, would also use a Li-polymer battery (~2000 mAh), yielding few more days of operation between recharges.

3.2.9. Sensors

The **Lifetch** ICU contains several standard sensors: acceleration, temperature, ambient light. The temperature and ambient light sensors perform important local functions. As the LCD contrast is inherently dependent on liquid crystal temperature, the driving voltage needs to be regulated accordingly to maintain display readability. The ambient light sensor is used to enable or disable display backlight to conserve energy.

Most recent readings from all those sensors are also available to the rule-based system (section 3.5.2). The use for accelerometers is obvious; the peak or instantaneous acceleration provides vital data about the ICU user's condition. The temperature sensor may provide warnings in case of dangerous decrease of temperature.

3.3. ICU user interface firmware

As the user interface is based on a touch panel display, the size of the menu is dictated by the human finger ergonomics. We have also taken into account the possibility of wearing gloves and working under adverse environmental conditions. Therefore we have focused on providing excellent readability. For instance, the display backlight is turned on automatically when the ambient light level is too low and the owner is using the touch screen. Additionally, if the ICU is not used for 15 s, the backlight is switched off to preserve battery life. Because of limited display surface (80x60 mm) we had to provide different "contexts". One of them contains only system status indicators. The other can display the map of the surrounding area; yet another one displays positions of the group members. The "HELP" button is available in all contexts. Messages can be displayed in every context.

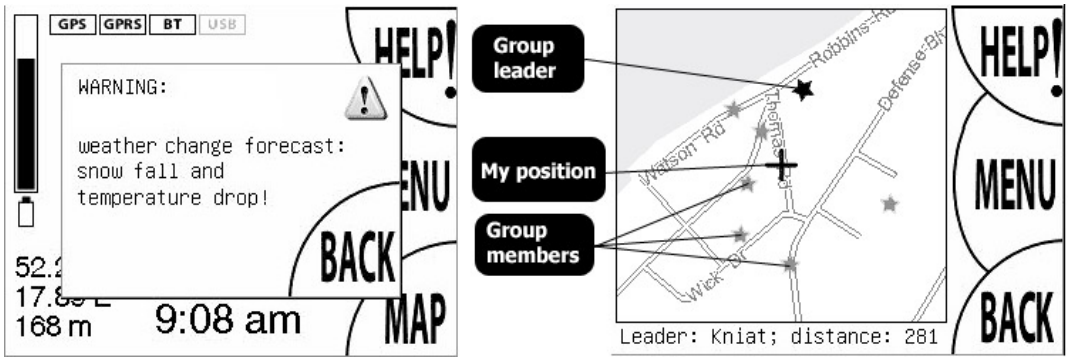
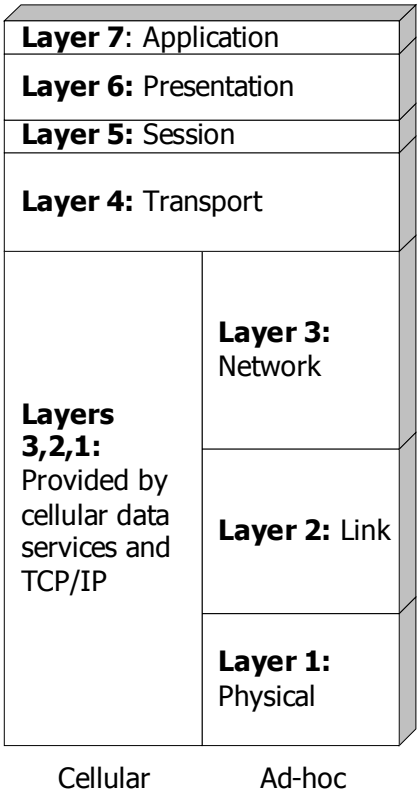


Figure 3.3 Examples of the ICU graphical interface

3.4. Communication protocols

While designing our communication protocols we chose the ISO/OSI reference model as a guideline. Our system takes advantage of two different network architectures: a cellular network (which is essentially a hierarchical network) and a peer-to-peer network (an ad-hoc mesh network). Thus, there are two different layer 3 (network) protocols. However, we decided that we should create one unified layer 4 (transport) protocol.



The presentation layer performs public key encryption functions (DES) and signatures to protect users' privacy.

Initially, we considered using the TCP for layers 4 and 5. However, as our address space does not match the IP address space and a complete TCP implementation is overly complex, we decided to use a customized version of TCP. We also added multicast functionality to layer 4.

Layer 3 protocol in the cellular network is very simple. A direct TCP/IP connection via cellular data service (e.g. GPRS) with the command center is established. This connection tunnels all layer 4 packets.

Layer 3 protocol in the ad-hoc network is more complex. We chose to implement a variation of the AODV algorithm [21]. It provides on-demand routing in dynamically changing networks. AODV has been shown to be highly scalable, unlike – for instance – Dynamic Source Routing (DSR).

Layer 2 (link layer) protocol implements guaranteed link-length delivery. The link layer uses Fire codes for error detection and correction. The retransmission is handled by the ping-pong ARQ protocol. It is possible to use more advanced schemes, like the window scheme, but we have found that with the low latencies inherent in short-distance radio links the memory overhead of window schemes would not be justified.

Layer 1 (physical layer) is not overly complex. The low transmission speeds (20-30 kbps) allow us to disregard the finite impulse response length of radio channel and a simple Manchester code is sufficient for data recovery. We operate in the unlicensed ISM band (around 915 MHz in the USA, 868 MHz in Europe).

Figure 3.4 The layered structure of the communication protocols

3.4.1. Application layer

Because there are many different messages that may be interchanged between devices in the application layer, we have chosen to standardize their formats using Abstract Syntax Notation One [3]. The packets are encoded in accordance with Basic Encoding Rules [4]. The binary form of packets is thus completely defined by the ASN.1 types. The basic message types are:

- **UpdatePacket** – message periodically sent by the ICU, it consists of all sensor data; position information from updates are saved in a FIFO-style log by all neighboring ICUs in case there is no access to the Command Center; the saved data can be retrieved from the ICUs by the Command Center to assist position prediction,
- **DemandReqPacket** – request for data (maps, weather, shortest path, etc.),
- **DemandRespPacket** – response to DemandReqPacket, containing requested data,
- **HelpReqPacket** – request for immediate help, consequence of pressing "HELP" button,
- **HelpAckRPacket** – message from the CC, sent when the rule-based system detects a potentially dangerous situation or the operator is concerned about the status of a certain ICU user; message meaning "Do you need help?",
- **HelpDeclPacket** – response of the ICU user to HelpAckRPacket, meaning "I am OK!",
- **MessagePacket** – message containing information of the types: emergency, warning, informative, commercial, group or user_to_user.

The detailed ASN.1 characteristic of the messages is shown in table below:

```
ApplicationLayer ::= CHOICE {
    update UpdatePacket,
    demandreq DemandReqPacket,
    demandresp DemandRespPacket,
    helpreq HelpReqPacket,
    helpackr HelpAckRPacket,
    helpdecl HelpDeclPacket,
    message MessagePacket
}
UpdatePacket ::= SEQUENCE {
    rule_id INTEGER,
    time UTCTime,
    pos GPSPosition,
    temp_1 INTEGER,
    temp_2 INTEGER,
    light_1 INTEGER,
    light_2 INTEGER,
    accel_x INTEGER,
    accel_y INTEGER,
    accel_z INTEGER,
    battery INTEGER
}
GPSPosition ::= SEQUENCE {
    longitude REAL,
    latitude REAL,
    altitude REAL
}
DemandReqPacket ::= CHOICE {
    map MapDemandReq,
    weather WeatherDemandReq,
    path PathDemandReq
    -- new types may be defined
}
MapDemandReq ::= SEQUENCE {
    pos GPSPosition,
    range REAL,
    sel_obj IA5String
}
WeatherDemandReq ::= SEQUENCE {
    pos GPSPosition,
    time UTCTime,
    range REAL
}
PathDemandReq ::= SEQUENCE {
    start CHOICE {
        pos GPSPosition,
        obj IA5String
    },
    end CHOICE {
        pos GPSPosition,
        obj IA5String
    }
}
DemandRespPacket ::= CHOICE {
    map MapDemandResp,
    weather WeatherDemandResp,
    path PathDemandResp
    -- new types may be defined
}
MapDemandResp ::= SEQUENCE {
    top REAL,
    bottom REAL,
    left REAL,
    right REAL,
    bitmap BIT STRING
}
WeatherDemandResp ::= SEQUENCE {
    warning BOOLEAN,
    text IA5String
}
PathDemandResp ::= MapDemandResp
HelpReqPacket ::= UpdatePacket
HelpAckRPacket ::= NULL
HelpDeclPacket ::= NULL
MessagePacket ::= SEQUENCE {
    class MessageClass,
    text IA5String,
    bitmap BIT STRING
}
MessageClass ::= CHOICE {
    commercial NULL,
    informative NULL,
    warning NULL,
    emergency NULL,
    group NULL,
    user_to_user NULL
}
```

3.4.2. Presentation layer (security issues)

During design phase we have put a strong emphasis on security and privacy issues. Particularly, sniffing and spoofing attacks should be prevented. In order to achieve this objective we have considered several encryption algorithms with symmetric and asymmetric keys in the presentation layer. Eventually we chose the 128 bit 3-DES algorithm because it is directly supported by our system controller and provides sufficient security for the requirements of our system.

A secret DES key is assigned together with the Id (unique within one operating area) to each ICU while it is being registered in the **Lifetch** system. It allows the ICU to encrypt messages addressed to the Command Center and decrypt messages from the Command Center.

In order to raise the level of security we decided to automatically change the ID and DES encryption key each time the ICU enters again the GSM range.

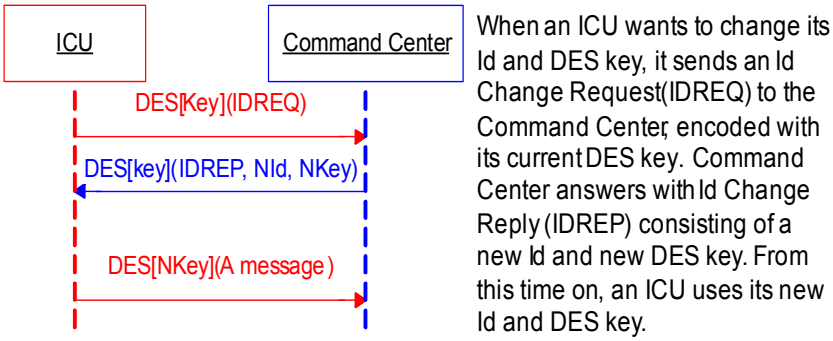


Figure 3.5 ICU Id and encryption key exchange protocol

The communications protocol, although a reliable one in sense of guaranteed delivery in case of intermittent link failures (thanks to TCP), is obviously subject to data loss because of permanent link failure. Therefore, the acquisition of new cryptographic keys by the ICU cannot be assured. Let us assume that the CC sends IDREP with a new key to the ICU and updates the key of this ICU in the database. However, if this packet does not reach the ICU, the ICU will not update its key. On the other hand, if the CC waits with the key update for the acknowledgement from ICU, it might not come even if the ICU received and updated its key. It can be proven that this problem is unsolvable, even in a synchronous model of distributed system (the proof is similar to the impossibility result for the Coordinated Attack Problem [22]). We have worked around this problem by keeping also the old key in the CC. If the currently active key does not decrypt the message, decryption with the old key is attempted.

3.4.3. Network Layer in ad-hoc network

Instead of creating a new routing algorithm from scratch, we took into detailed consideration several already verified source-initiated as well as table-driven algorithms for routing in ad-hoc networks [2]. Finally, we decided to use the Ad-hoc On Demand Distance Vector Routing (AODV) [21] tailored to our needs. AODV is source-initiated algorithm, which, in contrast to table-driven algorithms (such as the DSR), does not rely on periodic advertisements. This causes the demand on the overall bandwidth reachable to the mobile nodes (ICUs) to be substantially smaller, especially considering the fact that ICUs normally broadcast their packets every 15 minutes. Moreover, AODV is one of few algorithms supporting multicast, which is perfect for communication within a group. The detailed description of the AODV algorithm is widely available and will not be described in detail.

AODV (like other MANET routing algorithms) assumes that the message can only be delivered to the recipient when there is an instant route from the source to the destination. This is, obviously, insufficient for the needs of our system, because periodic ICU status update message retransmission is crucial for our system robustness. On the other hand, the message delivery time, although important, is not crucial. That is the reason why, in order to ensure that every single message will reach the destination (the CC), we decided to broadcast the message to all neighboring ICUs in case the AODV routing algorithm fails to establish a route to the CC. If the broadcast occurs the recipients store the position of the sender in a FIFO log. The flowchart of the sending process is shown in the figure below.

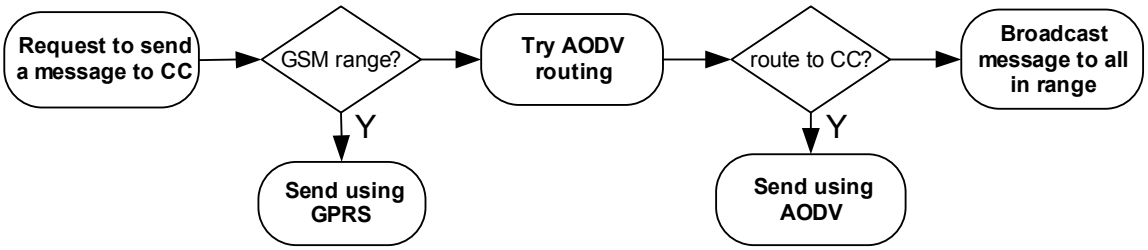


Figure 3.6 Block diagram of the message delivery channel selection algorithm

3.4.4. Link layer in ad-hoc network

Link layer frame:	Source address (32)	Destination address (32)	ACK flag	O/E flag	Packet length (8)	Packet data (184)	Fire EDC code (40)	Packet data (184)	Fire EDC code (40)	...

Each link layer packet is prefixed with a header. This header consists of source and destination addresses (32 bits each), an acknowledgement flag and a 1-bit sequence number. An 8-bit packet length field (in bytes) follows. Next, the packet data is sent. A 40-bit EDC (Error Detection and Correction) code is attached to the end of each 23-byte segment of a packet.

Each packet sent from unit A to unit B should be acknowledged by a packet from B to A with its acknowledgement flag set. The lack of acknowledgment results in the retransmission after 200 ms. After three failed retransmissions the packet is considered undeliverable due to link failure and a report is sent to the network layer.

The link layer also performs the MAC functions (Media Access Control). A CSMA (Carrier Sense Multiple Access) protocol is used for these functions. Units willing to transmit have to wait for at least 10 ms of unbroken radio silence. Each unit waits a random time, but less than 20 ms, after detecting silence.

EDC functions are implemented using Fire polynomial codes. These codes are optimized for correcting bursts of errors in a message. The main advantage of this solution in comparison with convolutional coding is a relatively low computational complexity.

3.4.5. Physical layer in ad-hoc network

Physical layer frame:	Preamble: 010101..01	Start flag: 11110000	Packet data

The physical layer is based on Frequency Shift Keying – the type of modulation in which the signal frequency is varied according to the currently transmitted bit, running at 19200 bps. We have applied Manchester encoding, which is directly supported by the system controller. Because synchronization is required for the physical layer, each link layer packet is prefixed with a variable-length preamble, which consists of alternating 1s and 0s and a 11110000 byte to designate its end.

Each system controller wakes up every 100 ms to check for new transmissions. It remains active for 500 ms, if a transmission was encountered, or for 30 ms, if there was no transmission within this time range. If a new packet arrives within this time, the activity timer is reset and another 500 ms will pass until the system controller switches into sleep mode.

If there was a transmission from this ICU in the last 500 ms, or a transmission was received with high power level it is clear that all nearby ICUs will be awake and a short preamble (64 bits) is sufficient. Otherwise, a 2048 bit long preamble is required to alert them.

3.5. Group cooperation

One of the most exciting features of **Lifetch** is the ability to bind ICUs into groups. Each ICU owner can create a new ICU group, becoming a Group Leader. The main function of the group of ICUs is to assure that members stay within a certain range from the leader and allow them to see the visualized positions of each other on the map. When one of the members exceeds the specified distance from the leader, the group leader receives a warning message on the screen of his/her ICU with the position of the member. Each group has a Group Id and Group Encryption Key. For security reasons the group creating procedure must be quite sophisticated. It is described in section 3.5.2.

3.5.1. Group management protocol

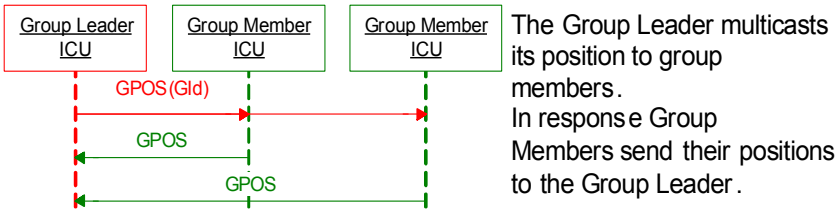


Figure 3.7 Sequence diagram presenting the exchange of position data in a group

3.5.2. The process of creating a group

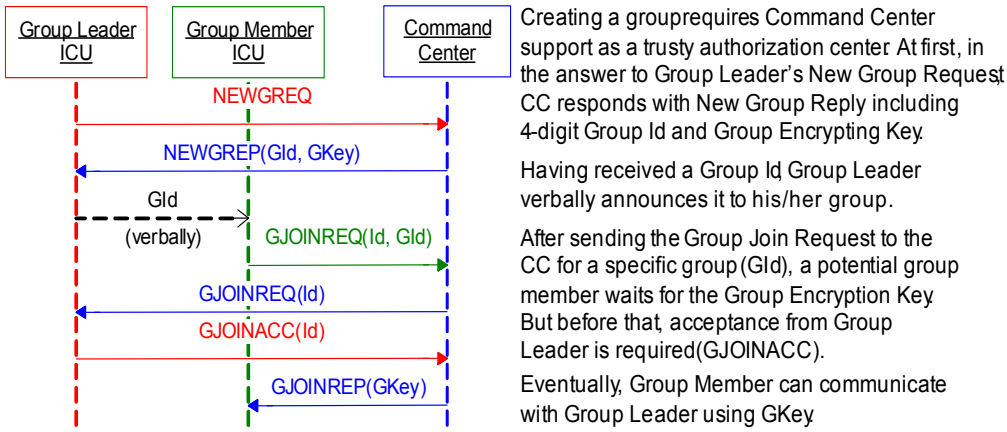


Figure 3.8 Sequence diagram presenting the process of creating a group

3.6. Command Center development

The Command Center is an application written using Visual Studio .NET. We applied a modular architecture, which helped us to work concurrently and will allow adding new features in the future. So far, we have implemented and tested subsystems crucial to provide the required functionality, which are the **Emergency Situation Detection Subsystem,** the **Position Prediction Subsystem,** the **Message Distribution Subsystem, On-demand Information Service Subsystem** and the **Administration Subsystem.**

Each module, as well as the whole Command Center, may be duplicated to provide the robustness that is crucial in a safety-related project.

All **Lifetch** data is held in a relational database. We considered using a native database interface, which is very effective, but does not assure sufficient portability. Considering this trade-off we chose the CC to connect to the database through ADO (ActiveX Data Objects), thus it is not dependent on any particular database management system (DBMS). This improves system scalability, because an end user can switch to more powerful DBMS in case one is needed. The only modification would be to change the data source. A part of the database diagram concerning the ICU sensor data, location history, group cooperation and structure of rules is shown below:

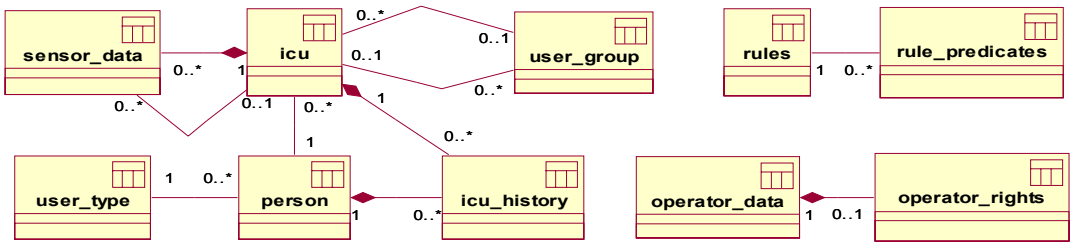


Figure 3.9 Data model diagram of the Lifetch database

In concern of users' privacy and system security we have designed a privilege-based security subsystem. Privileges can be set at two levels: DBMS specific (rights to view or modify data) and system level. On the system level the only one who may grant and revoke security privileges to/from other users is the operator with the "admin" privilege. System level privileges are held in the operator_rights table.

3.6.1. Map processing system

Maps of the protected area are an integral part of our system. The interface designed for the map processing is intuitive and allows the operator to zoom and scroll them. We could

create our own maps, but have chosen to use the MapPoint Web Service, that is providing those. The cooperation with the web service is done through a map provider. It is a Dynamic Link Library developed by us in C#. As long as the DLL provides an interface compatible with other modules of the CC, it could be easily replaced to change the source of maps to another web service or even a locally stored database. This way the system does not depend on any map supplier and operators can use their own maps if they possess more precise ones. The use of web service technology also makes it possible to divide required computational power among several computers. Thus the system is more scalable and can be set up on commonly available and cheap equipment.

Considering the map sizes and times required to retrieve them (e.g. from the web service), we decided to develop a caching mechanism which would improve overall system performance by storing often used maps in the local buffer. The map cache is placed in the map provider DLL, because this way we can take advantage of the particular map source's special features. It allowed us to transfer the map processing functionality to the map provider DLL and simplify the design and implementation of the Command Center.

3.6.2. Rule-based system

The intention of designing this subsystem was to automatically detect all kinds of potentially hazardous situations in the area under **Lifetch** protection. This subsystem obtains data from the main database and outside information sources such as weather web services. We designed a set of rules directly referring to the fields of the sensor_data table of the database. The initial idea was to periodically (e.g. every 15 minutes) run the complete set of rules for each sensor_data table tuple corresponding to one ICU. However, after performing several tests on a 10000 row table and a set of 25 quite sophisticated rules, we have discovered that processing these rules is too demanding computationally and does not meet our previous time requirements.

Eventually, we came up with an alternative, distributed-system approach – we divided the rules into 2 groups. The first group is responsible for detecting a dangerous situation based only on the state of a single ICU. The second group analyzes the global state of the system and reacts to the data received from outside services. We decided to shift the responsibility for processing Group 1 rules to the ICU device. This way, when a single ICU detects a potential threat based on its local sensor data, it immediately sends an 'update' packet to the CC, enclosing the rule identification number. It enables the CC operator to react much faster to the emergency, instead of waiting for its detection by the rule processing engine in the CC.

We have also developed a user interface for creating new rules. The operator has the choice to use a rule construction wizard or enter the rule manually which enables him/her to build more elaborate structures using multiple parentheses and function calls. When a new rule is added, it is compiled to a portable stack-machine p-code by the expression parser that we created. Then the rule is stored in the database and, if applicable (Group 1 rules), sent to all the ICUs during their next contact with the CC.

The benefits coming from introducing the rule based system may be presented by two scenarios describing a potentially dangerous situation and the reaction of the system.

1. The **Lifetch** operator receives information from the weather subsystem concerning a potential avalanche danger in a certain area under **Lifetch** protection. He raises the danger level for this area using the selection tool described in section 3.5.3. There is a pre-defined rule in the system:

   position.danger_level = 2 → send_msg(MSG_WARNING, sDangerName)

   The rule based system searches the database to find tuples that comply with conditional part of the rule and performs the corresponding action.

2. A hiker climbing a mountain slips on a rock and tumbles down a steep hill, hits a tree and loses consciousness. The value of peek acceleration during the fall is recorded by ICU.

This value activates the following rule from the ICU rule bank:

peak_accel > max_accel → send_msg(MSG_EMERGENCY, sRule, sParams)

and a message is sent to the Command Center operator, who is responsible for taking the appropriate action.

3.6.3. Message distribution

The purpose of this subsystem is to send messages to selected groups of ICUs. The messages can be sent automatically (as a result of operation of the rule-based subsystem) or manually by the operator. We have classified the messages into 4 basic types. The emergency and warning messages are used to alert the users to various hazardous situations, such as rapid weather breakdowns. The informative messages are location-aware and provide any type of data that can be useful to the user at a certain moment. The commercial messages are a business case concept, which assumes that the user agrees to periodically receive commercials, thus helping to finance the operating costs of the system. These types of messages are signaled in the ICU by different polyphonic ring tones. The user has the possibility to personalize his/her ICU to accept or reject some message classes and also set the polling frequency (how often does the ICU connect to the CC to check for new messages).

The operator can define a group of ICUs based on the location or information about the users stored in the database. The area selections are created by clicking on the displayed map and marking the consecutive vertices of the polygon. These polygons can be named and stored in the database for future use.

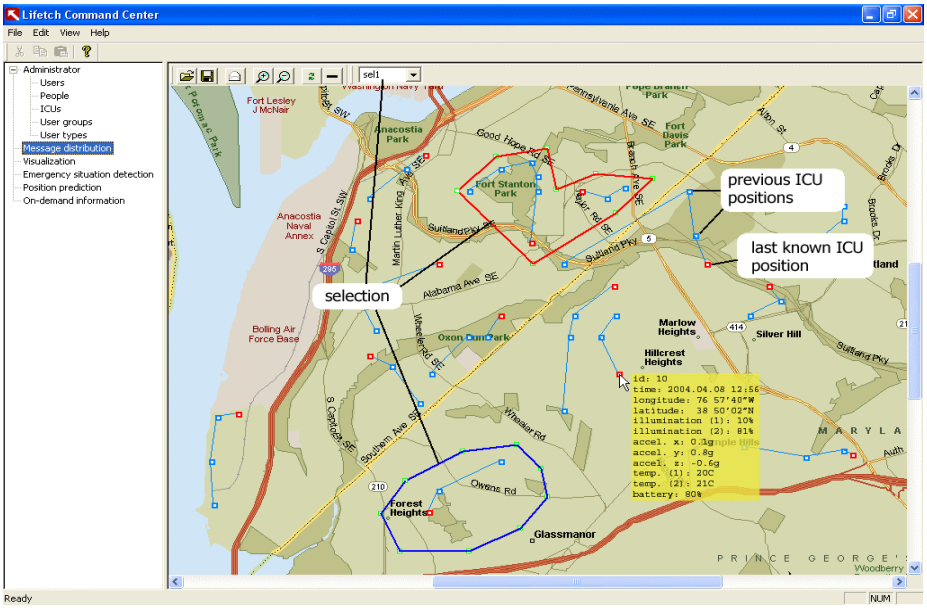


Figure 3.10 Screenshot from the Message Distribution Subsystem

3.6.4. On-demand information service

This system allows the ICU user to request various types of information. The most important functionality is downloading maps. The user can select the area directly, by specifying the map center point (latitude and longitude) and the dimensions in miles/kilometers. We have also developed an alternative way, where users can list the objects (their names or positions) they want on the map and the system will automatically choose the one that matches those needs best. Apart from that, the user may inquire about the weather forecast for his/her region, or ask about the list of nearest objects of a given type (our program supports all sorts of geographic objects, such as lakes, rivers or tourist facilities). There is also an option of receiving the shortest path to a given position or an object chosen from the list. The result is the map with the recommended path highlighted. When the Command Center receives an information request packet, it checks for the type and

properties of requested information, connects to an appropriate Web Service, fetches the data and sends it back to the ICU. The architecture of this system is open, allowing to extend it easily for example by adding a service providing information about historical background of places worth seeing nearby.

3.6.5. Position prediction subsystem

Position Prediction Subsystem (PPS) is a Command Center integrated program that assists the CC operator in narrowing down the area where a missing (either not responding for a preset time or reported missing by external sources) person can be found. This subsystem is based on area-specific information and on ICU location history, from which it derives the last position, average velocity and direction. After selecting the possible locations of the missing person, PPS visualizes them on the map. However, it is important to stress that PPS does not give 100% accuracy, but is a means of assisting the operator in making the right decision.

The PPS algorithm requires substantial knowledge about the area where **Lifetch** is operating. The operator can define certain area types to support the PPS system. By default, the whole area is divided into 3 categories. Some areas are marked as inaccessible (for example lakes), others are marked as rough terrain, difficult to walk through (for example a dense forest), and the other types are in the category of a standard area. Moreover, each area marked in such a way is assigned a maximum velocity that an ordinary person can reach in this area. The marking process is a job for an expert, who is familiar with the **Lifetch** operating area. It should be performed while setting up the system.

It should be noted that different classes of ICUs can be assigned different area types. For instance, while for hikers water areas are inaccessible, they may be good flying grounds for paragliders. The expert should prepare a different area classification for each ICU type.

The PPS algorithm we devised (which is a modification of a Markov chain being a probability-domain representation of possible ICU movements) consists of the following steps:

1. The area is divided into 100m x 100m squares,
2. Each square is assigned a default movement speed (0 for inaccessible squares),
3. Each square is assigned a number, which denotes the probability of finding the ICU in it; initially $s_{ij}(T=0)=\begin{cases}1 \; when \; i=i_0 \wedge j=j_0 \\ 0 \; when \; i\neq i_0 \vee j\neq j_0\end{cases}$, where i_0, j_0 – initial ICU position.
4. An initial transition probability of the ICU $t_{ij,n}(0)$ passing from each of the eight nearest neighbors to the square (i, j) is evaluated:
$$t_{ij,n}(T=0)=\left[|v_0|+\left(v_{0x}\cos\frac{n\pi}{4}-v_{0y}\sin\frac{n\pi}{4}\right)\frac{D_m}{\sqrt{(i-i_0)^2+(j-j_0)^2}+D_m}\right]\cdot v_{ij},\; n=0..7$$
where D_m – mean distance, where ICUs will probably change their initial movement direction, v_0 – initial ICU speed, v_{ij} – square movement speed.
5. A number of iterations are performed:
   - update state probabilities based on the transition probabilities:
   $$s_{ij}(T+1)=s_{ij}(T)+\left(\sum_{n=0}^{7}s_{i_nj_n}(T)\cdot t_{ij}(T)-\sum_{n=0}^{7}s_{ij}(T)\cdot t_{i_nj_n}(T)\right)\Delta s$$; Δs – discretization step.
   - update transition probabilities so that transitions which involve the ICU entering an area in which it is likely to be already placed is lessened (this way the simulated ICUs can avoid obstacles); Δt – discretization step:
   $$t_{ij,n}(T+1)=t_{ij,n}(T)+\left(t_{i_nj_n,-n}(T)\cdot s_{i_nj_n}(T)\cdot s_{ij}(T)-t_{ij,n}(T)\cdot s_{i_nj_n}(T)\cdot s_{ij}(T)\right)\Delta t$$, where
   $$i_n=\begin{cases}i-1 & when \quad n\in\{3,4,5\}\\ i & when \quad n\in\{2,6\}\\ i+1 & when \quad n\in\{0,1,7\}\end{cases},\quad j_n=\begin{cases}j-1 & when \quad n\in\{1,2,3\}\\ j & when \quad n\in\{0,4\}\\ j+1 & when \quad n\in\{5,6,7\}\end{cases}$$

j_n \ i_n	i-1	i	i+1
j-1	3	2	1
j	4	●	0
j+1	5	6	7

The final result is presented in a form of a state probability matrix *s*, which is then visualized on the map, as shown below. Variable transparency is used to increase readability.

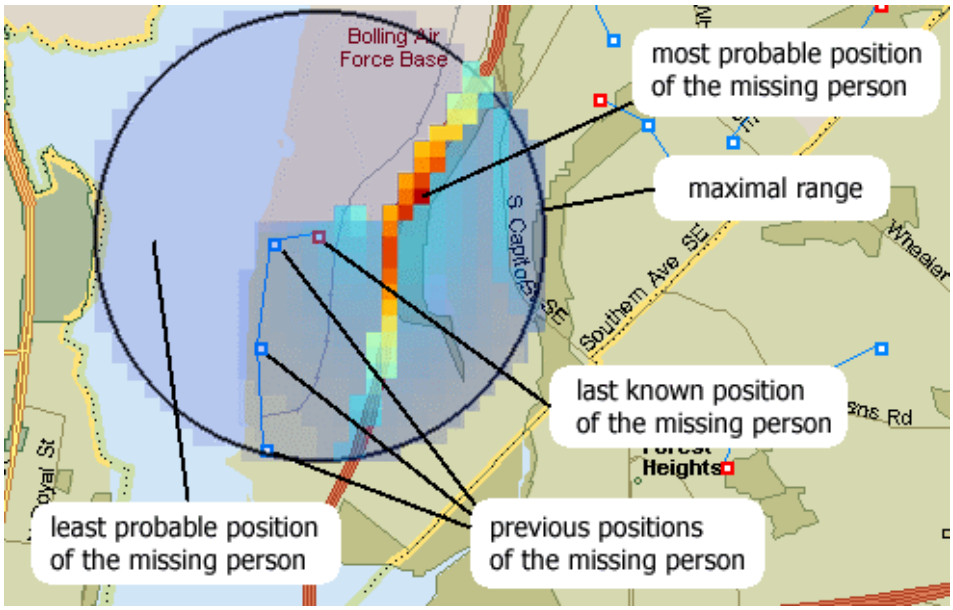


Figure 3.11 The effect of the position prediction algorithm

3.7. Tools developed

3.7.1 ICU simulator

In the phase of planning the project we have identified a critical path in our design process. The path consists of requirements identification, hardware design, prototype production, hardware testing, firmware development and testing. We decided to parallelize this process by assigning the task of writing an ICU simulator to the hardware designer. The firmware developer could test his software on this simulator before arrival of the hardware. The simulator also allowed easy low-level debugging and testing, which proved invaluable during operating system development.

The simulator is centered around the main processor. After in-depth studies of the SuperH datasheets and manuals the following approach was taken:

- write and verify an opcode-level simulator of the SH3 core,
- create MMU (Memory Management Unit) simulation module,
- create BSC (Bus State Controller) memory interface module,
- attach RAM and ROM simulators to the BSC and verify operation,
- implement the LCD controller subsystem,
- write serial port simulators,
- write CC1010 system controller module (includes the touch panel),

The multiple verification steps have been performed jointly by the hardware designer and the firmware developer. This intensive testing allowed us to become confident in the correctness of the operation of the simulator. In fact, the only error that has passed our initial verification was an error in the Renesas (manufacturer of the main processor) data sheets.

3.8. Verification and Validation

3.8.1 Command Center testing

While developing the Command Center, we conformed to the software engineering best practices in order to minimize the amount of errors in the source code. However, we decided that additional testing is necessary. In order to foresee the problems that a potential operator can face, we have decided on a black box approach. Obviously, the two persons involved in

writing the CC software could not participate in this kind of test. Consequently, the other two team members had to perform this task.

Upon completion of each CC subsystem they have been informed about the required functionality and asked to test the software for 3 days and report any errors or requested extensions. This proved especially helpful with the map processing and message distribution subsystems, where the test people helped to eliminate problems with map scale conversions and encouraged to add functionality to simplify the user interface. After redesign of these subsystems the group performed another set of tests.

We have also exploited the modular architecture of our software. Each module was thoroughly tested separately. For instance, we used the functionality provided by the MapPoint Web Service to perform comparative testing of the CC data conversion functions. The map cache was tested by repeatedly requesting randomly placed sections of maps and checking the hit rate, accuracy and program correctness. Similar approach was taken with other system modules.

3.8.2. ICU – ICU Communication testing

We have conducted many tests of all ad-hoc network layers. However, the fundamental results were obtained for the physical layer. The most important characteristic of the radio interface is the maximum attainable communications distance. However, as both link and transport layer protocols provide redundant retransmission capabilities, the rise of Bit Error Rate (BER) with distance causes gradual degradation of performance.

An acceptable Grade of Service (GoS) has been defined as no more than three link layer retransmissions in 100 packets. We have measured that this GoS in open hilly terrain corresponds to a distance of over 5 km at nominal transmission power of 0.8 W. At the maximum permissible power (100 mW) in Europe the range is over 2 km. In normal conditions the maximal range of the device is not utilized. We decided that the low network density achievable is not a justification for lowering the available bandwidth. This effect is coupled with increasing transmission range, as more ICUs have to share a radio frequency.

It has to be stressed that this range corresponds to one hop of the ad-hoc network. The total transmission distance can easily exceed several kilometers thanks to the multihop nature of the AODV routing protocol.

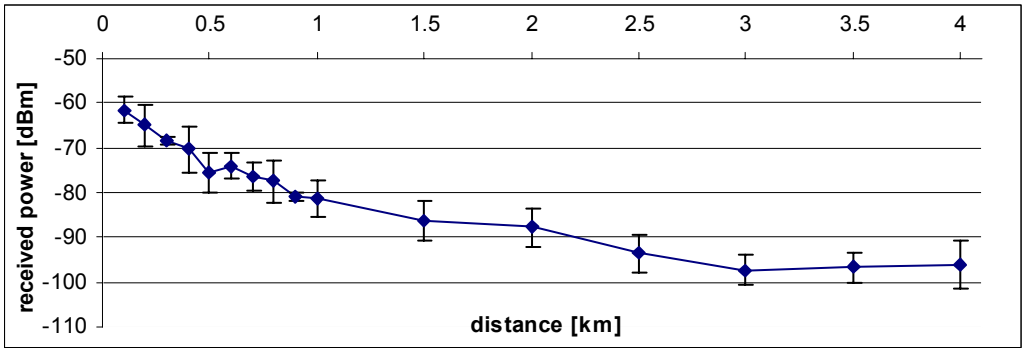


Figure 3.12 Received power as a function of distance at +30 dBm transmitted power

3.9. Costs

We have calculated the costs of a single ICU device to be as following:

- analog board (including parts) and battery – 35 USD,
- GPS receiver – 71 USD,
- digital board (including parts) – 48 USD,
- display and touch panel – 40 USD.

As we have built two such devices, the total spent amount is 388 USD.

4. Summary

We have designed a system that has rich technical capabilities and significant potential for saving human lives. It took almost 2 months for the vague idea of creating a safety-improving system for hikers in sparsely populated areas to evolve into the concept of **Lifetch** system, beneficial not only to hikers, but also to groups of tourists in cities or individual sightseers. We saw great potential in building a solution based on distributed personal units that are convenient to use, yet powerful and rich in functionality. Foreseeing the need for a system coordinating the work of these devices, we modeled the functions of the Command Center. During the 4 months of design and implementation, we have pursued new ideas, expanding the capabilities of the Command Center with a rule based emergency situation subsystem and enhancing the position prediction algorithms.

By now we have built two working prototypes of the ICU devices, which by complexity are equivalent to specialized palmtop. We have written over 4600 lines of C code for the firmware and over 14000 lines of C++ and C# code for the Command Center software, which has already been tested. With our firmware, the user can perform the following tasks:

- Signal an emergency and thus request immediate help.
- Browse the map of the nearby area and request further information about specific objects on it, or request a more detailed map of the nearby or another area.
- If the person is a member of the group, he/she can see the location of the rest of the group members on the map.
- If the person is a group leader, he/she can see the location of all the group members and monitor the status of their sensors.
- If a group member exceeds a previously set distance from the group leader, the leader is sent a warning message and he/she can multicast it to the group members.

We see many possibilities for improvement and further extensions, for example:

- Use of health monitoring of users (by monitoring pulse, pressure, or even performing EKG, the Command Center may suggest slowing the pace to accommodate the weakest member(s) of the group).
- One could tag camera images with location information obtained from the ICU, which in our context could be greatly helpful in rescue operations.
- Finally, the model – command center, a leader and a loosely coupled group – could be used for many other applications, notably military, but then the interaction between group members should be much stronger.
- We could take advantage of the existing satellite-based SARSAT system to broadcast help requests in the case when both GSM and ad-hoc networks fail.

There exists a question of marketability issues of our system. Location-based services, in general, do not have a clear business model, which slowed down their take-up. Emergency services (such as 911) are mandated by law and less related to business models and marketing. During business model analysis, we have found three ways of financing of the company operating a **Lifetch** system over a certain area. This company would profit mostly from renting the ICU devices. Firstly, it would be possible to sign contracts with an insurance company to lower the rates for users protected by **Lifetch**, thus encouraging them to rent the devices. Secondly, since the rescue action costs would be significantly reduced if **Lifetch** was introduced over a certain area, a part of the saved sum could be spent on the system's costs of operation. Finally, the users could be given an ICU device at a nominal charge, if they agreed to periodically receive commercial information during their trip, thus shifting part of the costs to outside companies. This commercial information might be even targeted at specific groups of users based on their preferences and observed behavior.

All in all, the system has great potential use possibilities and can provide significant safety improvements, making the world a safer place.

References

[1] Charles E. Perkins, Elizabeth M. Royer: Ad-hoc On-Demand Distance Vector Routing, 1999.

[2] Elizabeth M. Royer, Chai-Keong Toh: A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks, 1999.

[3] Abstract Syntax Notation One, ISO/IEC 8824-1, 2002.

[4] Basing Encoding Rules, ISO/IEC 8825-1, 2002.

[5] Jerzy R. Nawrocki, Bartosz Walter, Adam Wojciechowski: Toward Maturity Model for eXtreme Programming, IEEE Computer Press, Los Alamitos, 2001.

[6] Edward Yourdon, Death March: The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects, Prentice Hall, 1999.

[7] Les Besser, Rowan Gilmore: Practical RF Circuit Design for Modern Wireless Systems; Artech House, 2003.

[8] Commercial Semiconductor Components for RF/Wireless Applications, Alpha Industries, 1999.

[9] MAX2235 +3.6 V, 1 W Autoramping Power Amplifier for 900 MHz applications; Maxim Integrated Products, 2001.

[10] CC1010 Single Chip Very Low Power RF Transceiver with 8051-compatible Microcontroller; Chipcon AS, 2003.

[11] Krzysztof Wesolowski: Mobile Communication Systems, John Wiley and Sons, 2002.

[12] Man Young Rhee: Internet Security: Cryptographic Principles, Algorithms and Protocols, John Wiley & Sons, 2003.

[13] Joel Spolsky: User Interface Design For Programmers, Apress, 2001.

[14] Mohamed G. Gouda: Elements of Network Protocol Design, John Wiley & Sons, 1998.

[15] Gerard Holzmann: Design and Validation of Computer Protocols, Prentice Hall, 1990.

[16] Charles E. Perkins: Ad Hoc Networking, Addison-Wesley, 2001.

[17] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman: Compilers, Addison-Wesley, 1986.

[18] Scott Meyers: Effective C++, Addison-Wesley, 1997.

[19] Herb Sutter: Exceptional C++, Addison-Wesley, 1999.

[20] Jesse Liberty: Programming C#, O'Reilly & Associates, 2003.

[21] Charles E. Perkins, Elizabeth M. Royer, S. Das: RFC 3561, AODV Routing, 2003.

[22] Nancy A. Lynch: Distributed Algorithms, Morgan Kaufman Publishers, 1996.