



Tribhuvan University / Nepal
Institute of Engineering – Pulchowk Campus

TREMORFLASH

Earthquake Warning & Follow-Up System



TREMORFLASH

CSIDC 2004 Final Report
Making the World a Safer Place

Team Members AASHISH DUTTA KOIRALA
AMIT KRISHNA JOSHI
BISWA DAHAL
GIRISH BILAS JOSHI

Project Mentor ARUN K. TIMALSINA

CSIDC 2004

Computer Society International Design Competition

Department of Electronics & Computer Engineering
IOE Pulchowk Campus, Lalitpur, Nepal.

April 2004

1. Abstract

Earthquakes are a global phenomenon. They kill and cause harm to the human and animal population, and cause substantial amount of damage to property and infrastructure. We can neither prevent nor control earthquakes. The only protection one has from an impending earthquake is knowledge of the event – or any indication of its coming. So, constructing an earthquake warning and follow-up system fits in nicely with the theme *making the world a safer place*. Earthquakes are a cosmopolitan problem, and a solution in the interest of all people regardless of race, nationality or faith. Thus was born the **TremorFlash** system.

TremorFlash is a real-time system that can detect imminent earthquakes by sensing the primary waves that propagate before the deadly secondary waves arrive. A strategically placed central station detects earthquakes and broadcasts warning messages to all users in its coverage area. Users carry a portable module which sounds an alarm and lets them know about the coming earthquake. The users can then send back a reply of whether they are safe or need to be rescued back to the central station. Using this information, the central station, which maintains a database of all users, updates the users' records – which includes information about how they are (status) and where they are (GPS data). Thus, friends and family can query about their situation through a website. More importantly, rescue teams can use the website to receive a map of their area with potential victim “hot-points” mapped out. Thus, TremorFlash is a complete earthquake warning and follow-up system.

TremorFlash was built using an integrated approach towards software engineering and hardware organization. Object-oriented analysis and design was used and thus UML specifications and tools were used in order to design the system. Starting from requirements analysis using use case model, different tasks such as static modeling, behavioral modeling, implementation modeling and environment modeling was carried out. Finally, the design was implemented using robust high-level platforms. On the hardware front, the unavailability of hardware in Nepal was a major problem which required us to refine our requirements to match the available hardware – thus justifying our adoption of the spiral model.

Extra effort has been applied to make TremorFlash a reliable, safe and fault-tolerant system. Careful hazards and risk analysis was performed along with system dependability analysis in order to map out all risks involved with the system. Furthermore, a unique application instance redundancy technique was devised for fault tolerance.

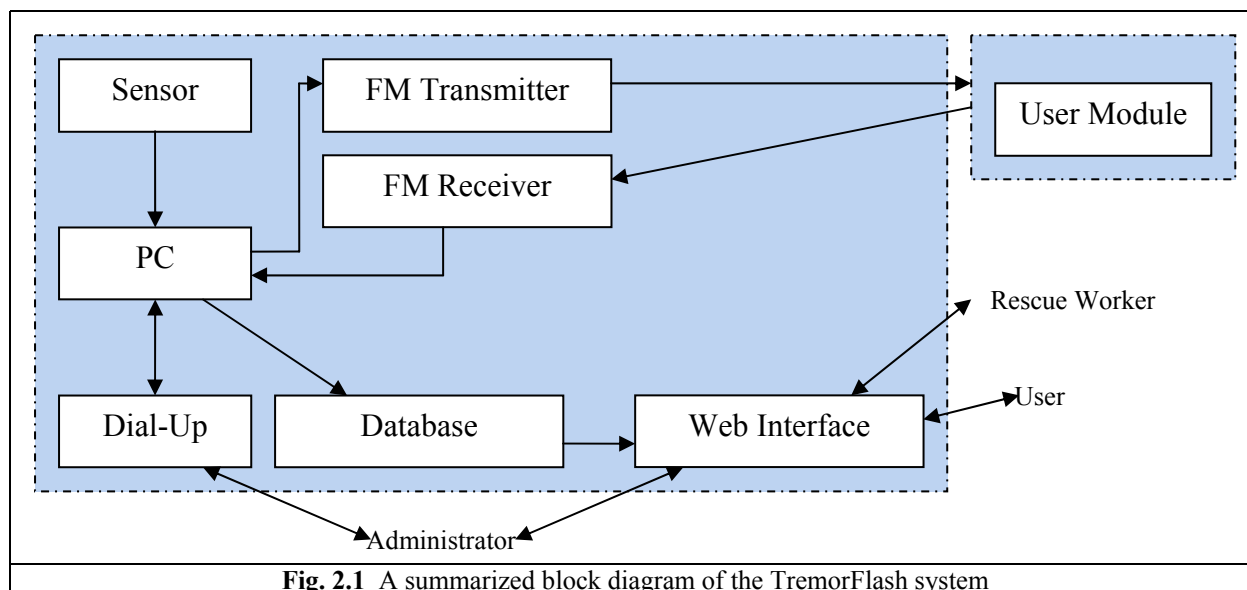
TremorFlash is unique and innovative in that it uses existing earthquake sensing technology that is relatively expensive for individuals to purchase, and provides a distributed service to numerous users. With existing quake sensors, the user has to buy the expensive unit and keep it at his/her home/workplace. In the event of an earthquake, an alert would generate only when the waves have reached the place itself. With TremorFlash, the user need only carry a portable unit, which sounds an alert when the earthquake waves are close to the origination point itself.

All these factors make TremorFlash an innovative and applicable project on world safety.

2. System Overview

2.1. System architecture

TremorFlash is a real-time system that can respond to an *aperiodic stimulus* (viz. the origination of an earthquake). It can detect imminent earthquakes, warn multiple users in its coverage area, and then follow up on the status of the users and map their locations so that they can be located easily by rescue teams. TremorFlash consists of two distinct subsystems: (i) TremorFlash core system (TFCS) and (ii) TremorFlash User Module (TFUM). A summarized block diagram is given below.



TFCS consists of a core system PC (CSPC) that is interfaced with an earthquake sensor, an FM transmitter module and an FM receiver module. It also maintains a database of all users in the coverage area. Parameters for CSPC can be changed remotely by the administrator using dial-up connection. A web interface is provided through which the administrator can control the database, rescue workers can map potential victims, and users can search for state of other users through a website. In the event of an earthquake, the alert signal from the sensor is processed and propagated to the transmitter by the CSPC. TFUM receives the alert signal and generates alarm for the user. The user can then send back his/her status to the TFCS which then logs this status on the database. The TFCS also sends back a reply to the user stating that his/her status has been received and logged.

2.2. Performance requirements

- **Performance of TFCS transmission:** The central transmission system should be reliable to operate in real time. Transmitting power should be within accepted government limits, but adequate to cover a small region accommodating 240 users, each with a bandwidth of 4kHz (inc. guard bands).
- **Performance of TFCS reception:** The central receiving system is housed in the same location as the central transmission system, and is required to be operating in real time and should be reliable.
- **Performance of TFUM:** Running with an independent DC power supply, it should consume less power, should be reliable, cheap and portable.

2.3. Design methodology

The system was built using the **spiral** model. **Object-oriented** analysis and design was used. The basic reason for the spiral model was that the requirements and the situations were constantly changing and the system needed to be capable of adapting to these changes gracefully. Key factors for such changes were mainly the availability of the required devices and the complexity of the system. A few examples of refinements and modifications that the system underwent under the spiral model are:

- Due to the unavailability of GPS chips, the system now uses a GPS receiver module found in GPS enabled laptops instead.
- A new component has been added that notifies the user that his/her status has been updated in the database after he/she responds to the earthquake alert.
- Trading in speed for simplicity and reliability, the continuous broadcast of initial alert bitstream is now handled by the software instead of using a shift-register circuit.
- Due to the unavailability of ISM based FM modules, the system now uses lower frequency transmitters and receivers.

Various tools and UML elements used during the design are listed below.

Table I – Development Tools

| Task | Model | Tool |
|-------------------------|-----------------------------------|----------------------------|
| Requirement analysis | Use case diagram | Microsoft Visio 2002 |
| Static modeling | Class diagram | |
| Behavioral modeling | State diagrams, sequence diagrams | |
| Implementation modeling | Component diagram | |
| Environment modeling | Deployment diagram | |
| Implementation | BASIC | Microsoft Visual Basic 6.0 |
| | JAVA | JCreator 2.4 |
| | PHP | PostNuke |
| | Assembly | Microsoft Assembler 5.1 |
| Project management | Gantt chart (for scheduling) | Microsoft Project |

2.4. Innovation and uniqueness

The major innovative aspects of our design are as follows.

- **Inexpensive and portable:** While conventional earthquake warning devices are expensive and are not portable, our design requires that the most expensive components reside in one place - the central transmission system, while the cheaper and portable warning devices are made available to a large number of people. This ensures a 1:N relationship between expensive-component and portable-cheap component (carried by each individual in a locality).
- **“Hot-point” mapping:** Since the user-module transmits users' GPS data to the central system, rescue workers can easily obtain "hot-points" in map for rescue operations. GPS, combined with wireless data transmission makes the system portable to use.
- **Works for other disasters also:** The system is designed such that it can be used as a generic disaster warning system and not just earthquakes. Using flood sensors, or fire-sensors, for example, the system can be converted to "Flood/Fire warning system" with few modifications if required. In other words, it forms a framework of any disaster warning system.
- **Easy rescue and query:** Web-interface allows rescue workers in different parts of the city/country/world to synchronize their efforts for efficient rescue operations. The web-interface also allows the public to query about the status of their family members whose whereabouts are not known during the event of the earthquake.

3. Implementation and Engineering Considerations

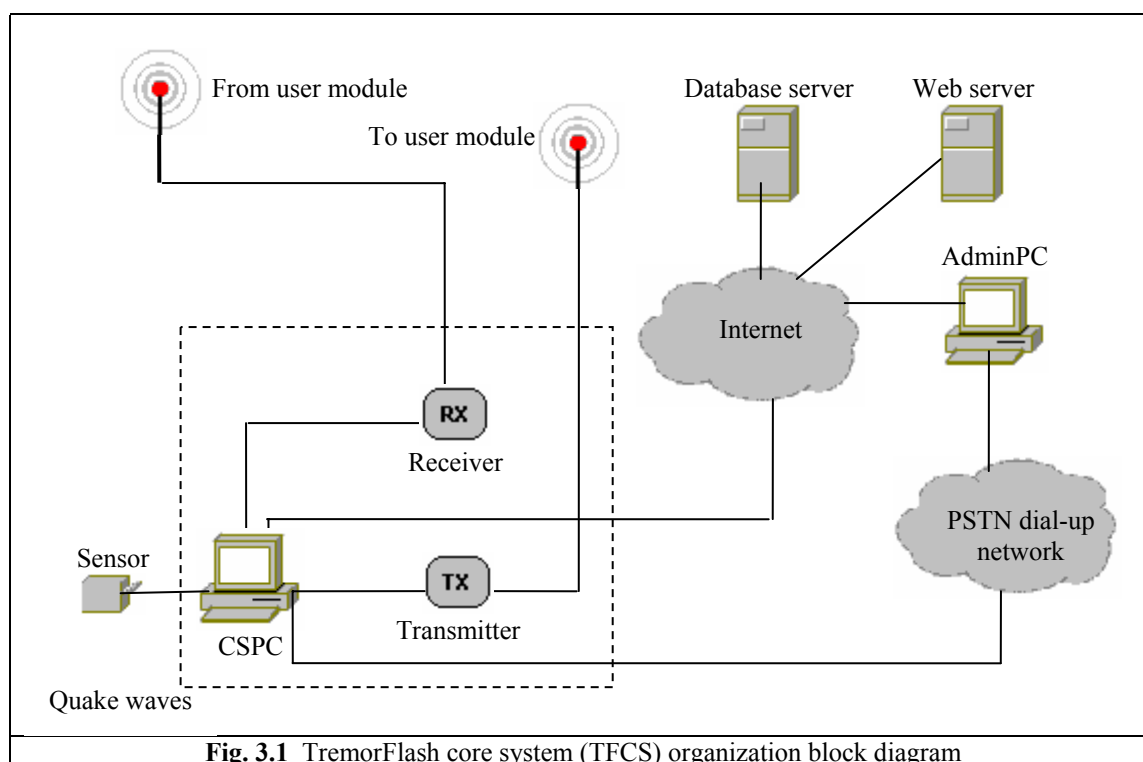
The TremorFlash system consists of two subsystems: (i) TremorFlash core system (TFCS), and (ii) TremorFlash user module (TFUM).

3.1. TremorFlash Core System (TFCS)

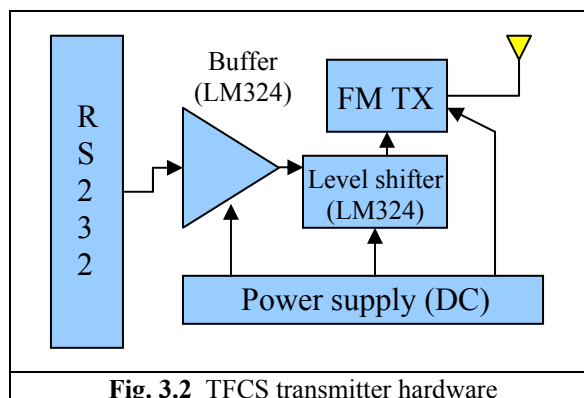
The TFCS serves the following functions:

- Maintaining a database of all users in the coverage area.
- Detection of signal from the sensor in the event of an earthquake.
- Broadcasting an alert message when the signal has been detected.
- Receiving user status from user modules, and updating database as required.
- Mapping the user's location using GPS technology for the aid of rescue workers.
- Providing reply to user modules informing them that their status is known to the system.

3.1.1. Hardware organization and operation



The above diagram shows the general TFCS organization. The TFCS hardware consists of: (i) sensor, (ii) transmitter and (iii) receiver. The sensor is connected to the core system PC (CSCP), which runs the TFCS software, via RS232 interface in simplex mode. When the sensor triggers a signal, an interrupt is generated to the CSCP. The CSCP is connected to the transmitter via RS232 interface in simplex mode as shown in the adjoining figure.



The receiver is also connected to the CSPC using a standard parallel port, as shown in the adjoining figure. The “sequential logic” block employs a sequential machine with 4 states (viz. a = 00, b = 01, c = 10, d = 11). The sequential machine is implemented using JK flip flops and is used to detect synchronization code in the received bitstream. Such a detection generates an interrupt, which initializes data transfer to the CSPC.

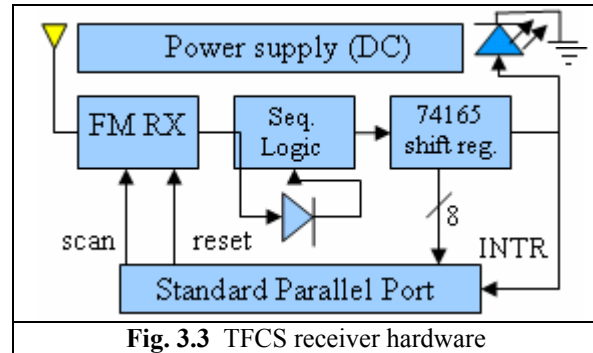


Fig. 3.3 TFCS receiver hardware

The selection of proper frequency level is done by the help of a SCAN line, which alters the center frequency of the receiver to the next ascending value, and the RESET line, which resets it to 88 MHz (the beginning range). The state diagram for the sequential machine is also shown alongside.

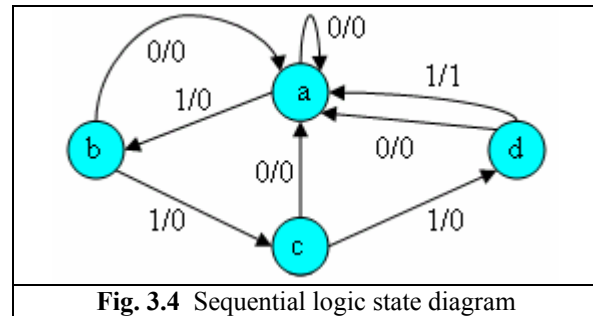


Fig. 3.4 Sequential logic state diagram

3.1.2. Software engineering

The software for TFCS was engineered using object-oriented analysis and design. Hence, the engineering process has been described using UML diagrams that accompany the text.

3.1.2.1. Requirements analysis and Use Case model

TFCS use case: The input from the sensor forms the basis for transmission earthquake alarm to the user. The user transmits a status signal to the TFCS, which logs the data into the database and provides acknowledgement to the user. Rescue teams interact with the Web interface, through which they coordinate rescue operation. The result of each rescue operation can be logged in by the rescue team to the database. The system administrator can configure the transmission and reception parameters, and administer database to add or remove users.

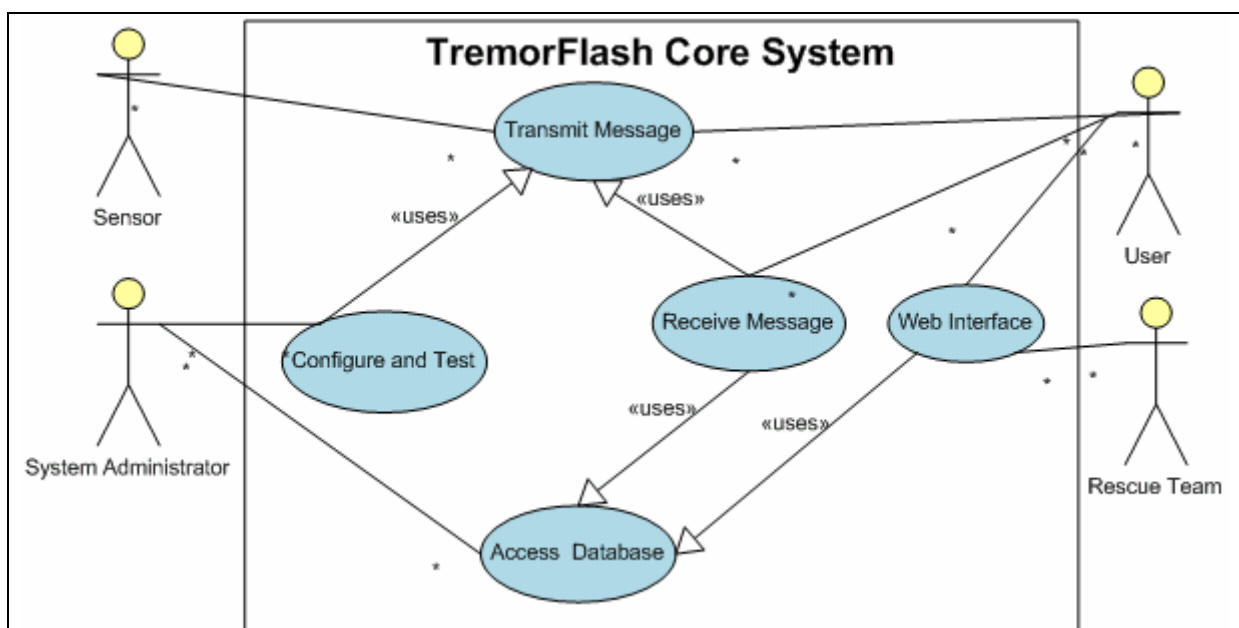


Fig. 3.5 TFCS use case diagram

3.1.2.2. Static model (Class organization)

The figure below shows the class diagram of the overall TFCS system. The user module (TFUM) is implemented using structured paradigm, and it does not have a UML representation.

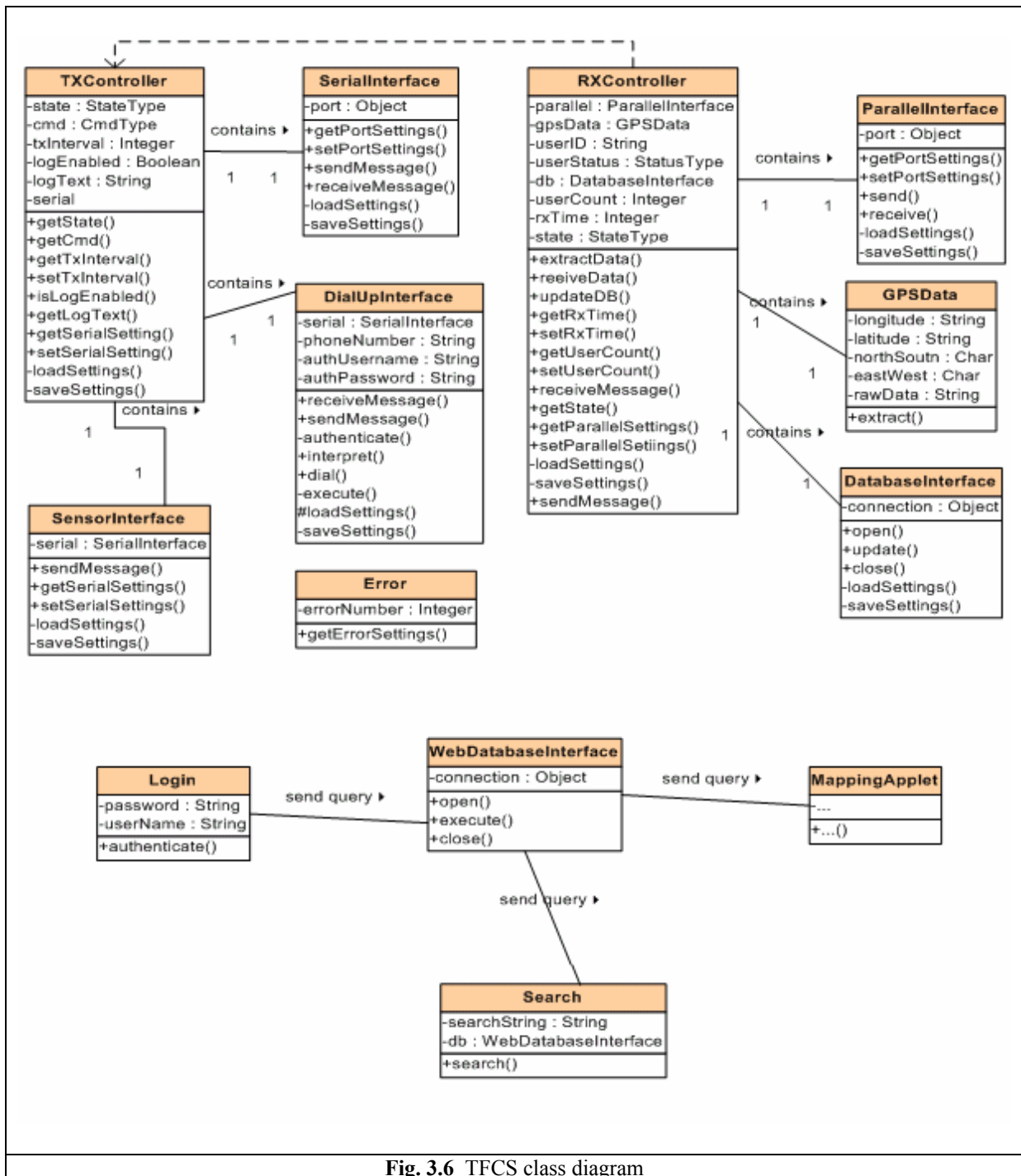


Fig. 3.6 TFCS class diagram

The TXController class forms the major component of the TFCS. This module interacts with other components to fulfill the following purposes.

Table II – Class Interaction Details

| Interaction | Purpose |
|-------------------------------------|---|
| <i>TXController-SensorInterface</i> | Sensor interfaces uses SendMessage to notify TXController to broadcast alarm signal. This alters the state of the system as shown in Fig. 3.8. |
| | Message value: MSG_BROADCAST_ALARM Message parameter: <i>Authentication code</i> – Read from the system registry. |
| <i>TXController-RXController</i> | RXController interface uses SendMessage to notify TXController to broadcast acknowledgement signal. Since the transmission signal only broadcasts messages to all users, the most appropriate form of acknowledgement signal to user is the user's ID, which is unique for each user. |
| | Message value: MSG_BROADCAST_ACK Message parameter: <i>UserID</i> – Read from parallel port by RXController. |
| <i>TXController-DialUpInterface</i> | Dialup interface used by the administrator, sets or retrieves configuration data of the TXController. These data include: 1. Transmission baud rate 2. Transmission time interval after which continuous transmission stop |
| | Message value: MSG_TX_CONFIG Message parameter 1: <i>GetSet</i> – zero implies query operation Message parameter 2: <i>Data</i> – Reference to variable of data for Get or Set method |

Class **SerialInterface** is implemented using MSCOMM32 ActiveX control in Visual Basic 6. Class **ParallelInterface** is implemented using PORTTALK.EXE, ActiveX EXE component for Visual Basic 6.

Class **DatabaseInterface** is implemented using ADO DB ActiveX type library in Visual Basic 6.

Table III – Classes for Web Interface

| Class | Implementation | Description |
|----------------------|--|--|
| Login | PHP | Used to authenticate rescue teams, and administrators. This feature is not available for normal users. |
| Search | PHP | Used to search users based on their names and user ID. This feature is available for any user of the system. |
| MappingApplet | PHP, Java Applet | Used by rescue workers to obtain an online map showing positions of casualties. |
| WebDatabaseInterface | PHP, with backend database using MySQL | Provides interface for the use of other classes in the web interface. |

3.1.2.3. Database design

The structure and design of the database system is illustrated by the following entity-relationships (ER) diagram.

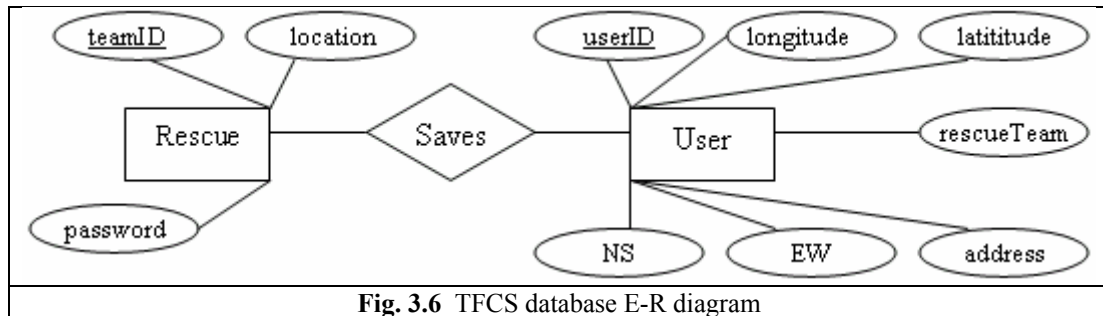


Fig. 3.6 TFCS database E-R diagram

The attribute definitions of the ER diagram are:

| Ta | Table Rescue |
|---|---|
| <ol style="list-style-type: none"> 1. <i>UserID</i>: A unique ID for each user. 2. <i>longitude</i>: GPS Longitude data 3. <i>latitude</i>: GPS latitude data 4. <i>NS</i>: North/South latitude 5. <i>EW</i>: East/West longitude 6. <i>address</i>: User address 7. <i>resuceTeam</i>: rescueTeam identifier currently involved in rescuing the user | <ol style="list-style-type: none"> 1. <i>teamID</i>: A unique identifier for each rescue worker team 2. <i>location</i>: Current location of rescue team 3. <i>password</i>: password for rescue team leader to access web interface |

3.1.2.4. Behavioral model

As explained above, the classes TXController and RXController have states that they go through. These are illustrated alongside in respective state diagrams.

Another component is the sequence diagram which is shown on the next page.

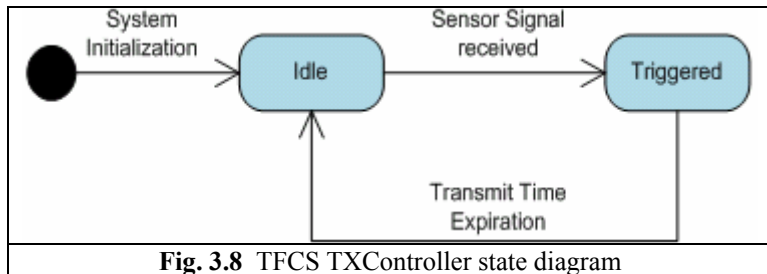


Fig. 3.8 TFCS TXController state diagram

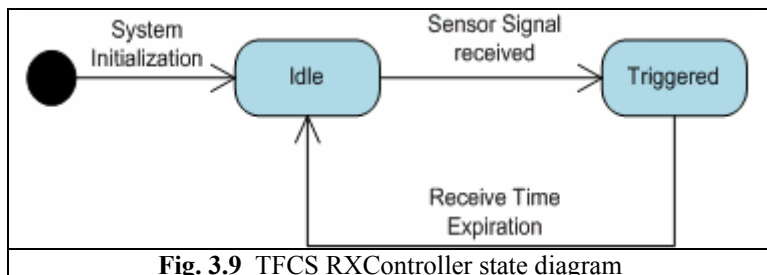


Fig. 3.9 TFCS RXController state diagram

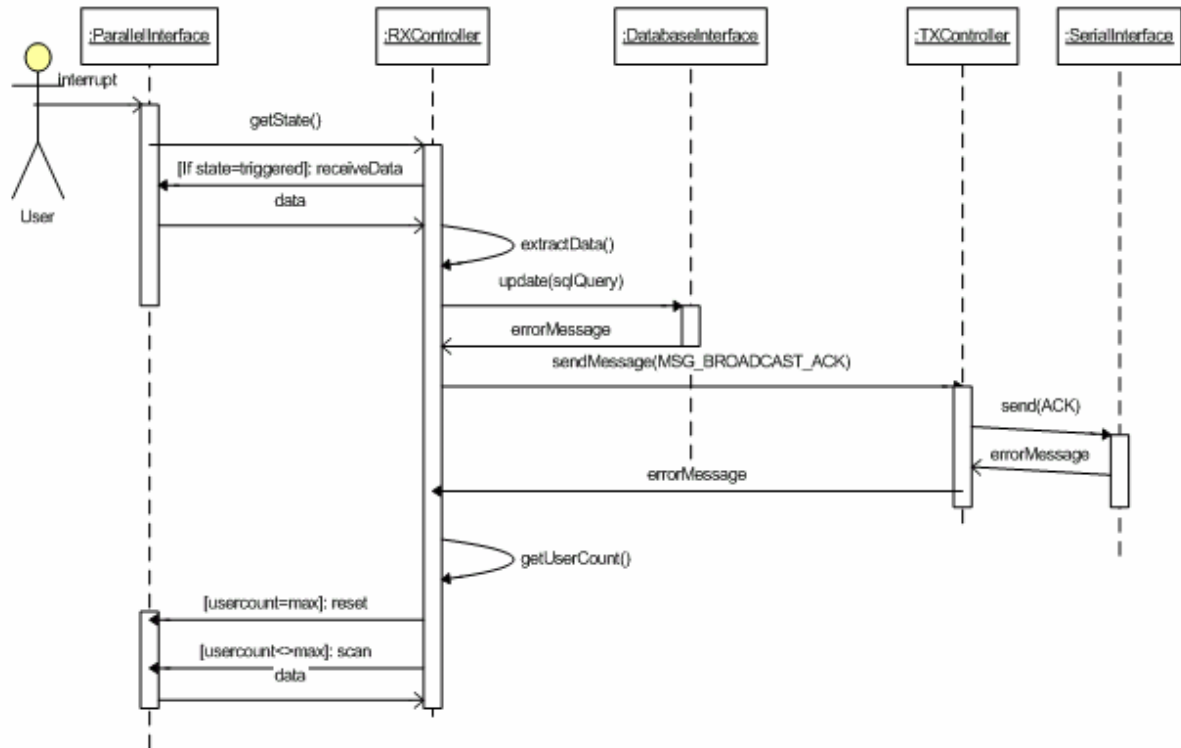
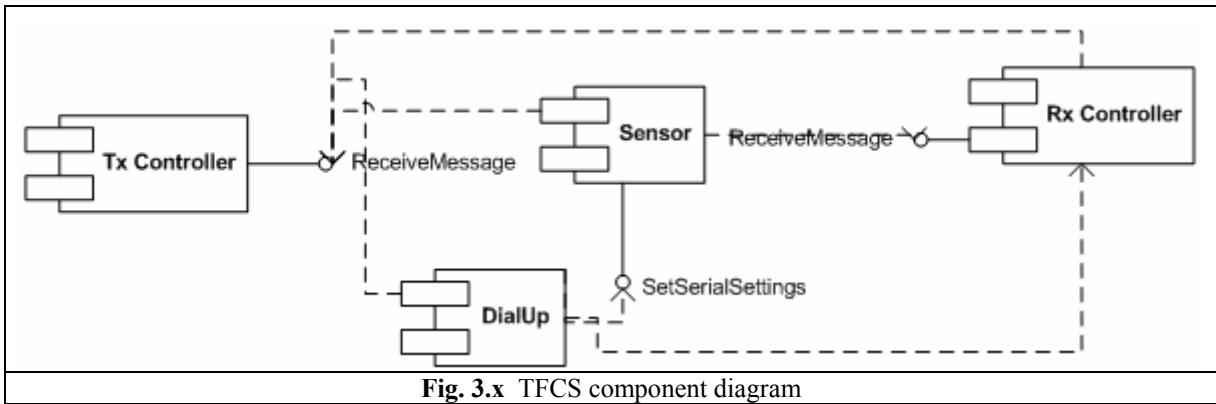


Fig. 3.x Sequence diagram for TFCS alert scenario

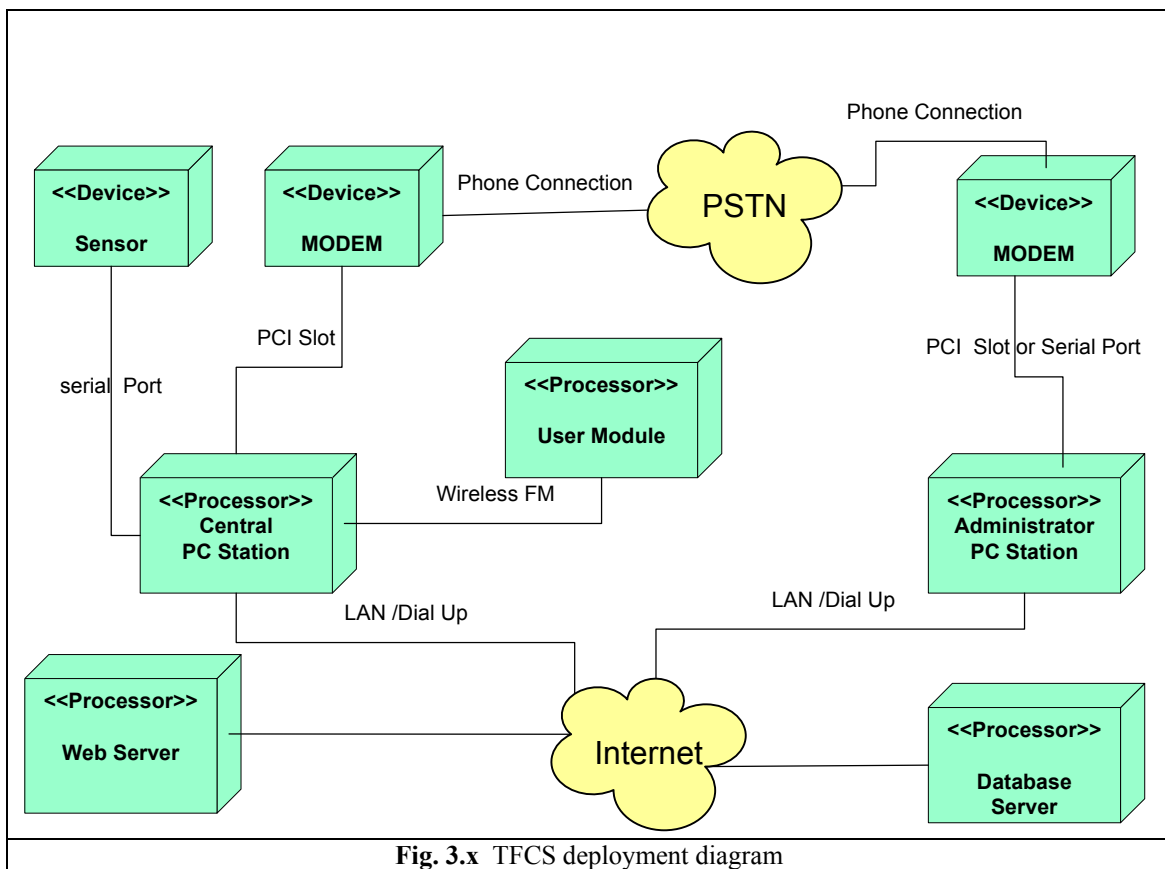
3.1.2.5. Implementation model

The TFCS software is implemented as components that communicate with each other. Four prominent components, viz. *TXController*, *RXController*, *DialUp* and *Sensor* have been modeled in the following component diagram.



3.1.2.6. Environment model

The TFCS software is deployed in the central PC station in the core system PC (CSPC), and also in the administrator’s terminal (AdminPC). The TFUM software is deployed in the user modules. The web interface and database systems are deployed in the web server and database server respectively. These various deployment nodes have been modeled below in the deployment diagram.

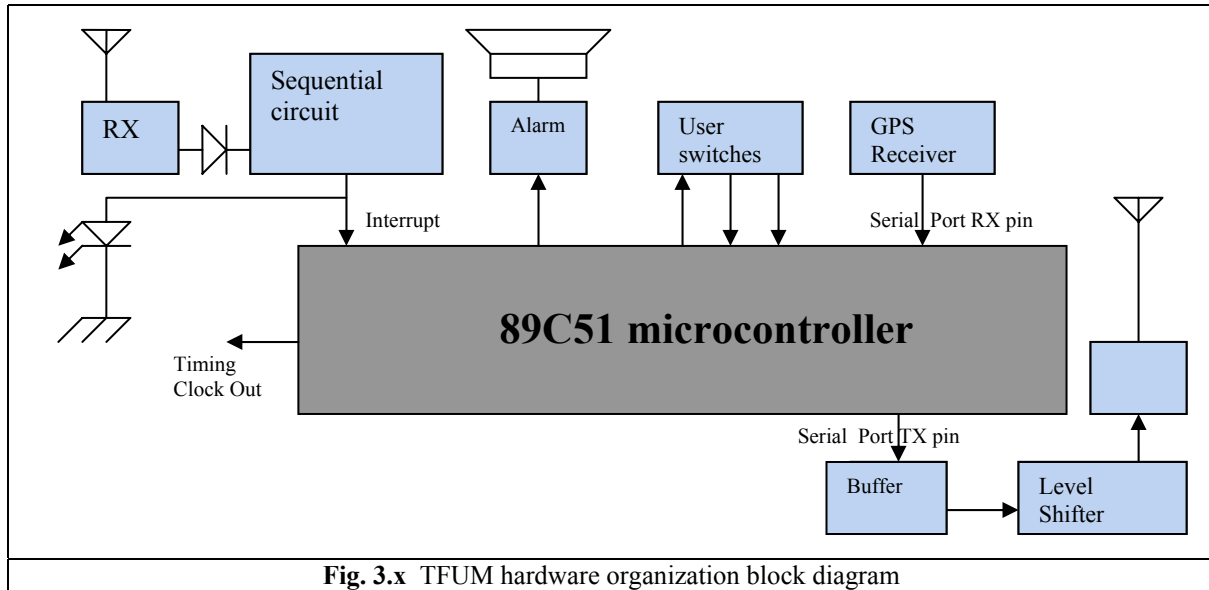


3.2. TremorFlash User Module (TFUM)

The TFUM is responsible for receiving alarm signal from TFCS and alerting the user, and allowing the user to report back his/her status to the TFCS.

3.2.1. Hardware organization and operation

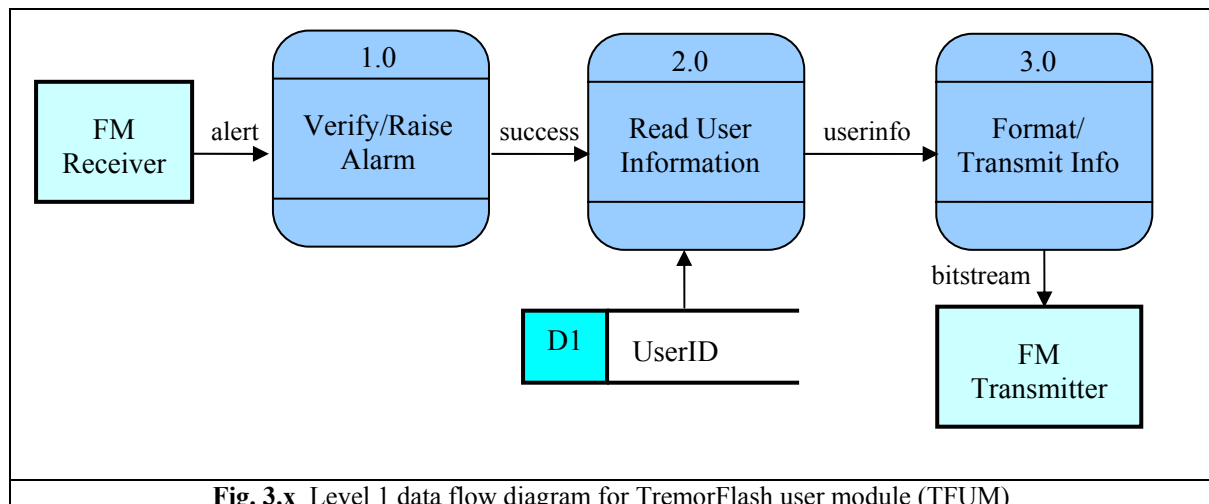
The user module hardware is summarized in the figure below. The sequential logic uses a state machine with the same architecture as described in the TFCS. Upon receiving a signal from the FM receiver, the state machine checks the synchronization code and extracts the data to be used by the TFUM software.



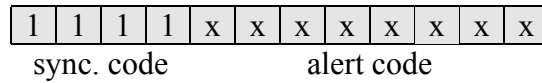
A non-maskable interrupt is generated to the microcontroller which runs the TFUM software. If the data is valid, the user is alerted through an audio alarm driver which drives a speaker. Push buttons marked SAFE and UNSAFE allow the user to report back his/her status to the TFCS. A serial port adapter allows data to be channeled to and from the transmitter and GPS receiver. Serial port has baud rate 9600 bps. Microcontroller clock operates at 11.059 MHz.

3.2.2. Software engineering

3.2.2.1. Data flow

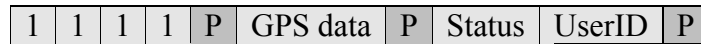


The above data flow diagram illustrates the flow of data through processes in the TFUM software. The *alert* data is a bitstream (12 bits) with the following structure:



The alert code is the authentication code for alarm generation, and the user ID for “status received” reply.

The *bitstream* data transmitted to the FM transmitter is 145-bits with the structure:



- 1111** Synchronization code
- GPS data** 16 bytes (8 bytes longitude + 8 bytes latitude)
- P** parity bit
- Status** 2 bits (01 = Safe, 10 = Unsafe, 00/11 = Unknown)
- UserID** 8 bits

3.2.2.2. Process flow

The flow of control and implementation of algorithms in the TFUM processes outlined above are illustrated by the following flowcharts.

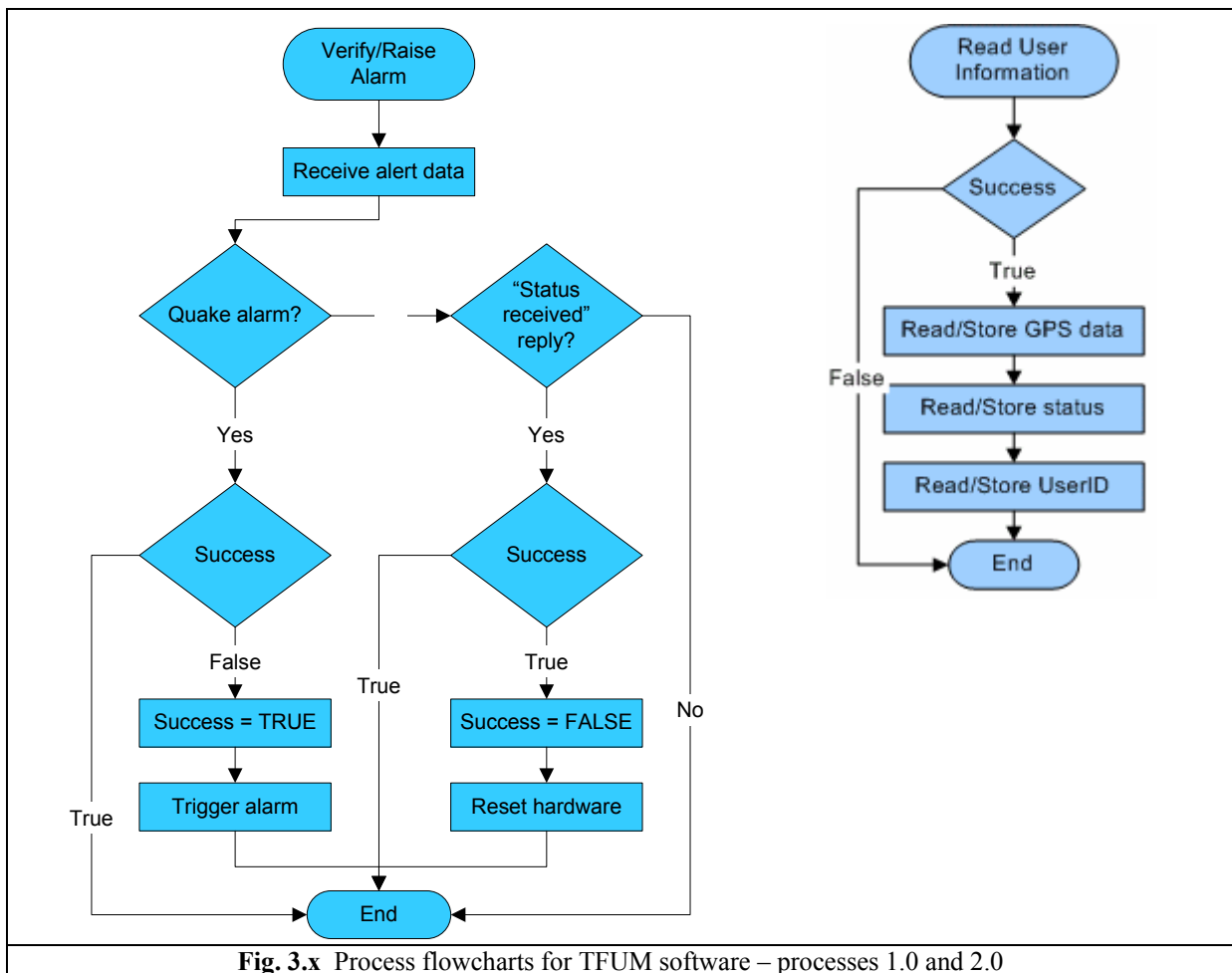
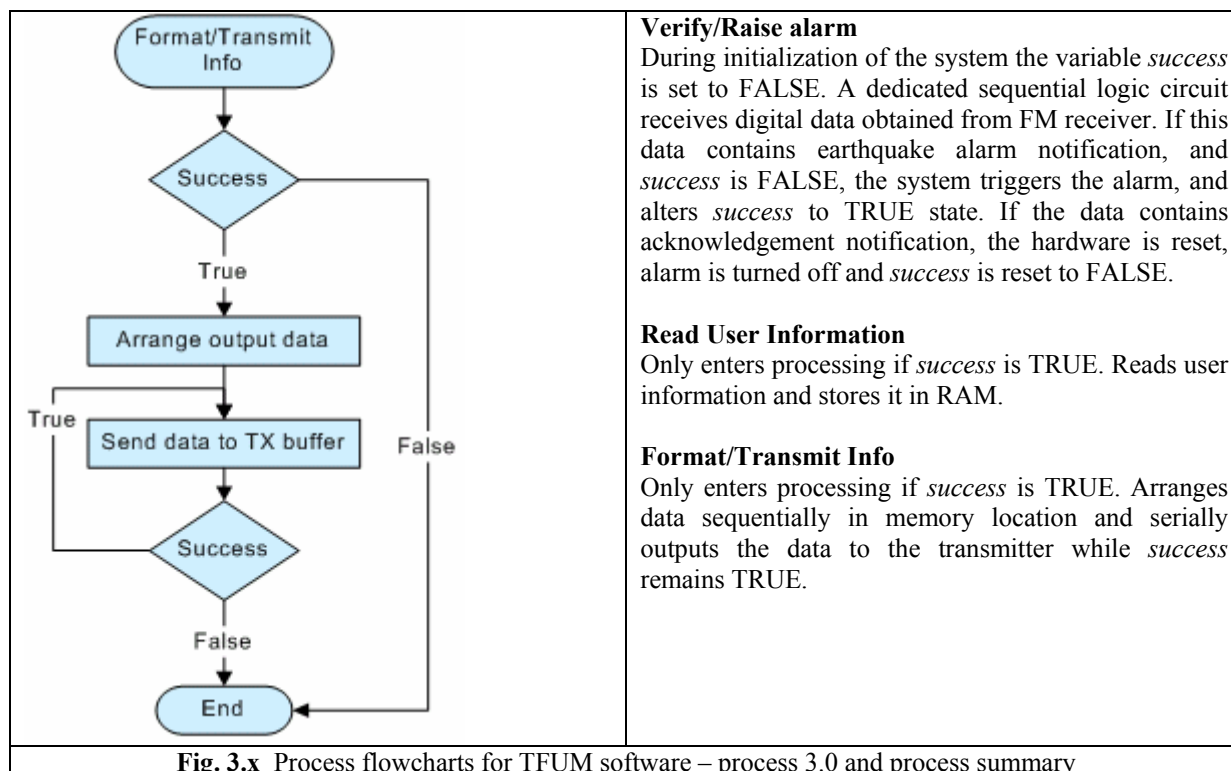


Fig. 3.x Process flowcharts for TFUM software – processes 1.0 and 2.0



3.3. Risk analysis and system dependability

3.3.1. Hazard and risk analysis

The analysis phase of the project involved analysis of various hazards. Possible hazard reduction steps were formulated after risk analysis. This is summarized by the table below:

Table IV – Hazard and risk analysis

| Identified Hazards | Probability of Hazard | Severity of Hazard | Risk | Acceptability | Risk Reduction Step |
|---|-----------------------|--------------------|--------|---------------|-------------------------------|
| Sensor Error | Medium | High | High | Intolerable | Use reliable sensors |
| Data transmission error from user module to central station | Medium | High | High | Intolerable | Interleaved parity check bits |
| Data transmission error from central station to user module | Medium | Low | Low | Acceptable | Repeated transmission |
| GPS Receiver Error | Low | Medium | Low | Acceptable | Use reliable receivers |
| Database system malfunction | Low | High | Medium | ALARP | - |
| Rescue team infidelity | Low | High | Medium | ALARP | - |

3.3.2. Dependability

TremorFlash is a dependable system. The four criteria of dependability of the system is summarized below.

Table V – Dependability of the system

| Aspect | State | Imple |
|---|---|---|
| Availability | Yes | Operational Full time |
| Reliability 1. Fault tolerance 2. Transmission Error Detection 3. Human Error Risk | Yes Yes Low | Software fault tolerance discussed below Parity Checking Minimal user intervention. Only expert human professionals interact with critical parts of the system. |
| Safety | Yes | Following the risk reduction steps as pointed out in Hazard Analysis section above. |
| Security 1. Denial of Service 2. Data Corruption 3. Sensitive information disclosure | Possible Possible Low possibility | - Parity checking. Non-sensitive information in database. |

3.3.2.1. Software fault tolerance

Software fault tolerance is implemented by running two identical processes together, one of which monitors the other. Only one of these processes remains active in central transmission and receiving system. This active process handles all the essential core activities of TFCS. However, the other process monitors the active process without any participation in TFCS. This model is shown using the following diagram.

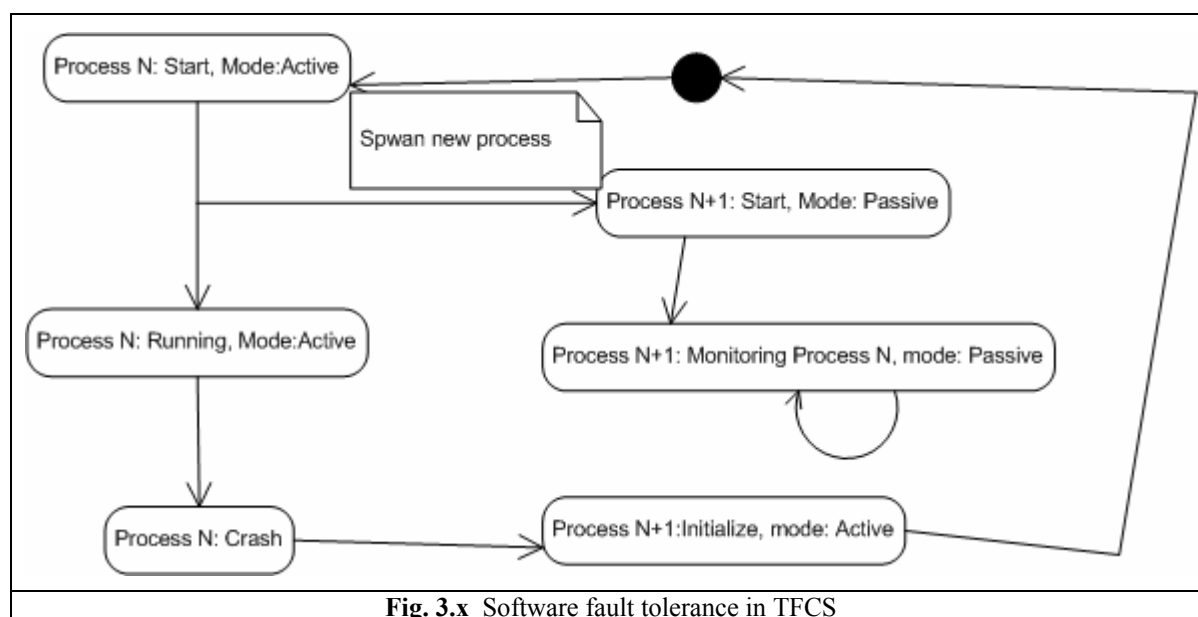


Fig. 3.x Software fault tolerance in TFCS

The outcome is that there is always an instance of the software running – even if, during the course of operation, an instance of the software crashes.

3.4. System tradeoffs

In order to optimize the design and to accommodate the changing requirements, some tradeoffs had to be considered. While these tradeoffs might have reduced the efficiency of the system in some areas, we believe that the benefits gained outweigh the efficiency lost.

- Our modular approach resulted in increased system cost. However, the result of the increased modularity is that the system is more user-friendly, configurable and maintainable. It also more fault-tolerant and less prone to failures.
- The addition of the component responsible for informing the users that the database has been updated was carried out later in the project schedule. Although this somewhat increased the complexity of the system, it resulted in better service for the user and more user-friendliness.
- We decided to minimize the use of hardware and maximize the use of software as far as possible. We felt that software control is more feasible and more configurable, and that implementation of complex hardware was prone to unpredictability and was less configurable and more difficult to develop. The use of dedicated hardware would certainly have made the system faster and more efficient, though.
- Finally, in order to accommodate the fact that specialized hardware was mostly unavailable, we had to refine the system design so as to use alternative hardware.

3.5. Verification and testing

Various tests performed during the course of development of hardware and software components for TremorFlash are summarized below.

Table VI – Verification and testing

| T | Test Procedure/Result |
|-----------------------------|--|
| TFCS transmission component | Sensor signal was emulated using a closed circuit, feeding into the RS 232 port. The transmitting hardware in serial port (COM 2) was initialized. Receiver, at a distance of 5 m from the laboratory received the signal. |
| TFCS reception component | Square wave signals fed from function generator, at a rate of 9.6 kHz, and connected to parallel port, caused glowing of LED, marking the detection of synchronization of code. This was immediately followed by detection of signals in the user module FM receiver circuit. FM signals transmitted from user modules, were occasionally not detected. This was found to be due to the shift in center frequency of the transmitting module. |
| Dial-up interface module | This software module was successfully tested to configure transmission specific data in the TremorFlash Core System (TFCS). Messages to alter the transmission interval from 20 s to 5 s in: TXController, and RXController modules was successfully reflected in application data value changes in the system registry. This software module was used to send a “TEST” command, to the central system from the a remote PC, connected through a phone line via modem, the user module detected the transmission, however did not raise the alarm. This conforms with our expected operation. |

3.6. Project management

| Sn | Description | Wks | Dec. 2003 | | | | Jan. 2004 | | | | Feb. 2004 | | | | Mar. 2004 | | | | Apr. 2004 | | | |
|--------------------------------|------------------------------------|-----|-----------|---|---|---|-----------|---|---|---|-----------|---|---|---|-----------|---|---|---|-----------|---|---|---|
| | | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| PHASE I : WARN | | | | | | | | | | | | | | | | | | | | | | |
| 1 | Sensor Interface | 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | FM Tx interface | 1 | | | | | | | | | | | | | | | | | | | | |
| 3 | FM Transmitter | 4 | | | | | | | | | | | | | | | | | | | | |
| 4 | Central System Low level software | 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | Central System High level software | 8 | | | | | | | | | | | | | | | | | | | | |
| 6 | FM Receiver | 4 | | | | | | | | | | | | | | | | | | | | |
| PHASE II : WHERE AM I ? | | | | | | | | | | | | | | | | | | | | | | |
| 1 | Sequential logic | 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | User Interface Hardware | 1 | | | | | | | | | | | | | | | | | | | | |
| 3 | GPS Receiver Interface | 12 | | | | | | | | | | | | | | | | | | | | |
| 4 | FM Transmitting Interface | 1 | | | | | | | | | | | | | | | | | | | | |
| 5 | FM Transmitter | 1 | | | | | | | | | | | | | | | | | | | | |
| 6 | Microcontroller Software | 8 | | | | | | | | | | | | | | | | | | | | |
| 7 | FM Receiver | 1 | | | | | | | | | | | | | | | | | | | | |
| PHASE III : SAVE ME | | | | | | | | | | | | | | | | | | | | | | |
| 1 | Sequential Logic | 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | FM Receiver Interface | 4 | | | | | | | | | | | | | | | | | | | | |
| 3 | On Site PC Low Level Software | 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | Database Implementation | 2 | | | | | | | | | | | | | | | | | | | | |
| 1 | Website Development | 6 | | | | | | | | | | | | | | | | | | | | |

4. Summary

With TremorFlash, what has been attempted to achieve is an object-oriented approach, reliability, fault tolerance and portability and economy for the user. The team believes that with what has been created so far, these goals have been achieved. A real-time system has been designed and implemented that is capable of responding to aperiodic stimuli such as earthquakes. Most of the system is ready and is undergoing testing.

One component that is in progress is the mapping module which is responsible for using GIS technology to map GPS coordinates onto a visual map. This service is meant to be used by rescue workers to point out victim “hot-points” in their areas and if possible, also the location of individual victims. The mapping module was bound to take more time due to its programming intensive nature. It should be ready by the time it needs to. The rest of the system is ready and may be regarded as a working prototype.

There is a lot of room for improvement in the system, mainly in two ways. First, the system itself can be made more robust, more reliable and more fault-tolerant. Second, useful new features can be added to the system in order to make it even more efficient and applicable to real world problems. Along the first path, the database could be horizontally fragmented and distributed for redundancy and efficiency. A user database is bound to be large and should not be kept in its whole and in one place. Obviously, better transmitters and receivers are an immediate prospective improvement. Along the second path, one key fact is that this system does not have to be limited to earthquakes. It can be expanded to include any disaster that can be detected before it strikes.

The team believes that TremorFlash is quite manufacturable, marketable and maintainable. The required devices and units can be packaged very cost-effectively, mass-produced and sold. To maintain the system, however, cooperation from corporations and governments becomes a key issue. For the system to be truly effective, stations must be distributed globally covering strategically chosen regions. This is clearly not possible without the cooperation of governments and also big corporations. However, the public interest factor, the team believes, makes it a lucrative investment for most big corporations and a noble decision for most governments and government agencies.

Finally, a word on the relevance to the theme *making the world a safer place*. There are quite a lot of threats that undermine safety in the world today. Natural disasters have been around the longest. They are very hard to avoid or predict. Earthquakes, especially, cannot be controlled like landslides or avalanches can. So, the only way one can feel safe from earthquakes is if he/she knows when one is going to strike. Whatever little time the warning gives users to prepare could prove valuable and be the difference between life and death. Thus, we believe TremorFlash does contribute towards making the world a safer place.

5. References

5.1. Literature References

1. *Software Engineering: A Practitioner's Approach*, Roger S. Pressman
2. *Software Engineering*, Ian Sommerville
3. *Modern System Analysis and Design*, J.A. Hoffer, J.F. George, J.S. Valacich
4. *Microelectronic Circuits*, Sedra and Smith
5. *Digital Communication*, Simon Haykin
6. Manuals and datasheets for various hardware components used

5.2. Online References

- MSDN Library for Visual Studio 6.0
- Java 2 SDK Documentation from java.sun.com
- PHP Manual from www.php.net
- MySQL reference from www.oreilly.com
- www.virtualearthquake.com
- www.windmill.co.uk/gps.html
- www.smartdraw.com

5.3. Personal References

1. Prof. Dr. Dinesh Kumar Sharma, Department of Electronics & Computer Engineering, IOE Pulchowk Campus
 2. Prof. Dr. Prem Nath Maskey, Department of Civil Engineering, IOE Pulchowk Campus
 3. Prof. Jagan Nath Shrestha, Department of Electronics & Computer Engineering, IOE Pulchowk Campus
 4. Dr. Subarna Shakya, HOD, Department of Electronics & Computer Engineering, IOE Pulchowk Campus
 5. Dr. Rajendra Shrestha, Department of Mechanical Engineering, IOE Pulchowk Campus
 6. Assoc. Prof. Shashidhar Ram Joshi, Department of Electronics & Computer Engineering, IOE Pulchowk Campus
 7. Assoc. Prof. Timila Yami Thapa, Department of Electronics & Computer Engineering, IOE Pulchowk Campus
 8. Madan Kaji Shakya, Director, New Services Directorate, Nepal Telecom Ltd.
-