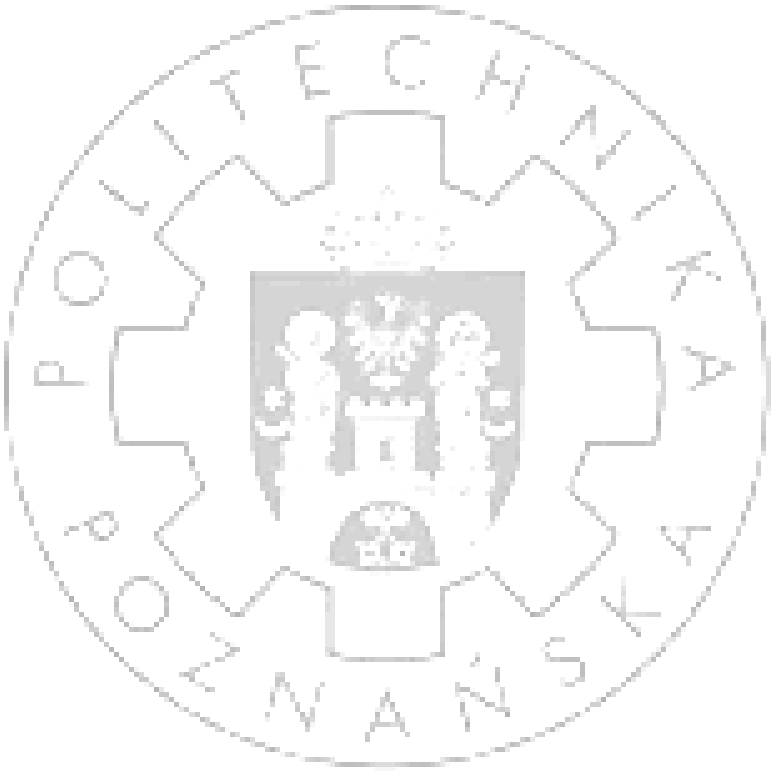
BlueEyes Conscious Brain Involvement Monitor

Poznań University of Technology

Application ID: 4RJND3

Region ID: 8



Poznań University of Technology

Institute of Computing Science

Team members	Paweł Kowalik	paulo@blueteam.wox.org
	Piotr Kubiaczyk	piotrq@blueteam.wox.org
	Krystian Nowak	zak@blueteam.wox.org
	Stanisław Osiński	stachoo@blueteam.wox.org
	Tomasz Pużak	tom@blueteam.wox.org
Project mentor	Jan Kniat, Ph. D.	Jan.Kniat@put.poznan.pl
Head of department	Jacek Błażewicz, Ph. D., Dr. Habil., Professor	blazewic@sol.put.poznan.pl

1. Abstract

Human error is still one of the most frequent causes of catastrophes and ecological disasters. The main reason is that the monitoring systems concern only the state of the processes whereas human contribution to the overall performance of the system is left unsupervised. Since the control instruments are automated to a large extent, a human – operator becomes a passive observer of the supervised system, which results in weariness and vigilance drop. This, he may not notice important changes of indications causing financial or ecological consequences and a threat to human life. It therefore is crucial to assure that the operator's conscious brain is involved in an active system supervising over the whole work time period.

It is possible to measure indirectly the level of the operator's conscious brain involvement using eye motility analysis. Although there are capable sensors available on the market, a complex solution enabling transformation, analysis and reasoning based on measured signals still does not exist. In large control rooms, wiring the operator to the central system is a serious limitation of his mobility and disables his operation. Utilization of wireless technology becomes essential.

BlueEyes - the system developed by our team is intended to be the complex solution for monitoring and recording the operator's conscious brain involvement as well as his physiological condition. This required designing a Personal Area Network linking all the operators and the supervising system. As the operator using his sight and hearing senses the state of the controlled system, the supervising system will look after his physiological condition.

2. System overview

BlueEyes system provides technical means for monitoring and recording the operator's basic physiological parameters. The most important parameter is saccadic activity[1], which enables the system to monitor the status of the operator's visual attention along with head acceleration, which accompanies large displacement of the visual axis (saccades larger than 15 degrees). Complex industrial environment can create a danger of exposing the operator to toxic substances, which can affect his cardiac, circulatory and pulmonary systems. Thus, on the grounds of plethysmographic signal taken from the forehead skin surface, the system computes heart beat rate and blood oxygenation.

The **BlueEyes** system checks above parameters against abnormal (e.g. a low level of blood oxygenation or a high pulse rate) or undesirable (e.g. a longer period of lowered visual attention) values and triggers user-defined alarms when necessary.

Quite often in an emergency situation operators speak to themselves expressing their surprise or stating verbally the problem. Therefore, the operator's voice, physiological parameters and an overall view of the operating room are recorded. This helps to reconstruct the course of operators' work and provides data for long-term analysis.

Our system consists of a mobile measuring device and a central analytical system. The mobile device is integrated with Bluetooth module providing wireless interface between sensors worn by the operator and the central unit. ID cards assigned to each of the operators and adequate user profiles on the central unit side provide necessary data personalization so

[1] Saccades are rapid eye jumps to new locations within a visual environment assigned predominantly by the conscious attention process.

different people can use a single mobile device (called hereafter DAU – Data Acquisition Unit). The overall system diagram is shown in Figure 1.

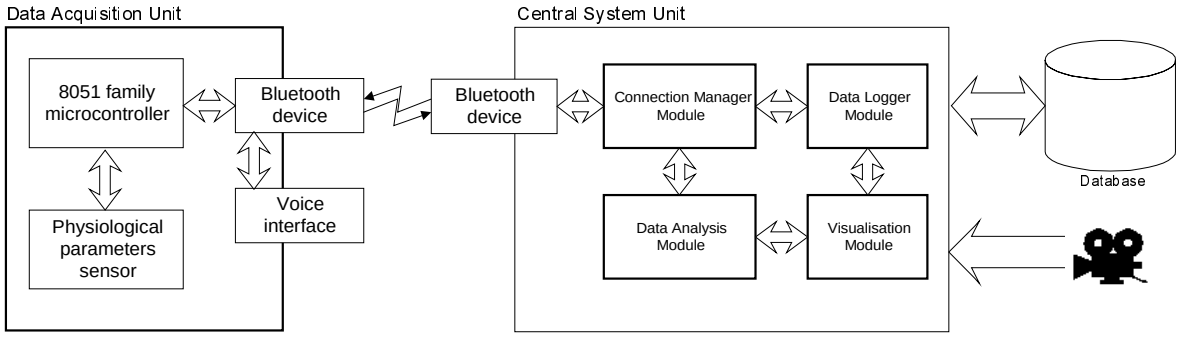


Figure 1. Overall system diagram

The tasks of the mobile Data Acquisition Unit are to maintain Bluetooth connections, to get information from the sensor and sending it over the wireless connection, to deliver the alarm messages sent from the Central System Unit to the operator and handle personalized ID cards. Central System Unit maintains the other side of the Bluetooth connection, buffers incoming sensor data, performs on-line data analysis, records the conclusions for further exploration and provides visualization interface.

Performance requirements

The portable nature of the mobile unit results in a number of performance requirements. As the device is intended to run on batteries, low power consumption is the most important constraint. Moreover, it is necessary to assure proper timing while receiving and transmitting sensor signals. To make the operation comfortable the device should be lightweight and electrically safe. Finally, the use of standard and inexpensive ICs will keep the price of the device at relatively low level.

The priority of the central unit is to provide real-time buffering of incoming sensor signals and semi-real-time processing of the data, which requires speed-optimized filtering and reasoning algorithms. Moreover, the design should assure the possibility of distributing the

processing among 2 or more central unit nodes (e.g. to offload the database system related tasks to a dedicated server).

Design methodologies

In creating the **BlueEyes** system a waterfall software development model was used since it is suitable for unrepeatable and explorative projects. During the course of the development UML standard notations were used. They facilitate communication between team members; all the ideas are clearly expressed by means of various diagrams, which is a sound base for further development.

The results of the functional design phase were documented on use case diagrams (Fig. 2, 3 and 4). During the low-level design stage the whole system was divided into five main modules (Fig. 1 – thick frames). Each of them has an independent, well-defined functional interface providing precise description of the services offered to the other modules. All the interfaces are documented on UML class-, interaction- and state diagrams. At this point each of the modules can be assigned to a team member, implemented and tested in parallel. The last stage of the project is the integrated system testing.

Innovative ideas

The unique feature of our system relies on the possibility of monitoring the operator's higher brain functions involved in the acquisition of the information from the visual environment. The wireless link between the sensors worn by the operator and the supervising system offers new approach to system overall reliability and safety. This gives a possibility to design a supervising module whose role is to assure the proper quality of the system performance. The new possibilities can cover such areas as industry, transportation (by air, by road and by sea), military command centers or operating theaters (anesthesiologists).

3. Implementation and engineering considerations

3.1. Functional design

During the functional design phase we used UML standard use case notation, which shows the functions the system offers to particular users. We defined three groups of users: operators, supervisors and system administrators.

Operator is a person whose physiological parameters are supervised. The operator wears the DAU. The only functions offered to that user are authorization in the system and receiving alarm alerts. Such limited functionality assures the device does not disturb the work of the operator (Fig. 2).

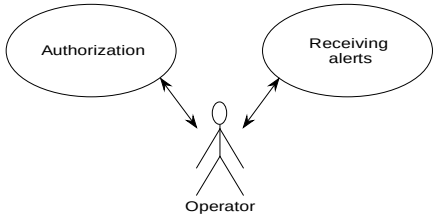


Figure 2. Mobile device use cases

Authorization – the function is used when the operator's duty starts. After inserting his personal ID card into the mobile device and entering proper PIN code the device will start listening for incoming Bluetooth connections. Once the connection has been established and authorization process has succeeded (the PIN code is correct) central system starts monitoring the operator's physiological parameters. The authorization process shall be repeated after reinserting the ID card. It is not, however, required on reestablishing Bluetooth connection.

Receiving alerts – the function supplies the operator with the information about the most important alerts regarding his or his co-workers' condition and mobile device state (e.g. connection lost, battery low). Alarms are signaled by using a beeper, earphone providing central system sound feedback and a small alphanumeric LCD display, which shows more detailed information.

Supervisor is a person responsible for analyzing operators' condition and performance. The supervisor receives tools for inspecting present values of the parameters (*On-line browsing*) as well as browsing the results of long-term analysis (*Off-line browsing*).

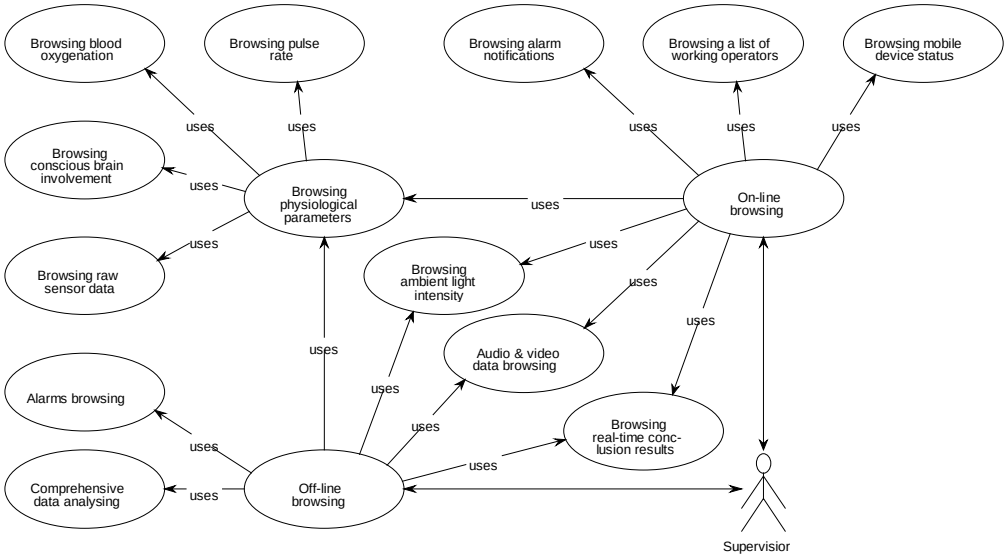


Figure 3. Functionality offered to the supervisor

During the on-line browsing it is possible to watch a list of currently working operators and the status of their mobile devices. Selecting one of the operators enables the supervisor to check the operator's current physiological condition (e.g. a pie chart showing active brain involvement) and a history of alarms regarding the operator. All new incoming alerts are displayed immediately so that the supervisor is able to react fast.

However, the presence of the human supervisor is not necessary since the system is equipped with reasoning algorithms and can trigger user-defined actions (e.g. to inform the operator's co-workers).

During off-line browsing it is possible to reconstruct the course of the operator's duty with all the physiological parameters, audio and video data. A comprehensive data analysis can be performed enabling the supervisor to draw conclusions on operator's overall performance and competency (e.g. for working night shifts).

System administrator is a user that maintains the system. The administrator is delivered tools for adding new operators to the database, defining alarm conditions, configuring logging tools and creating new analyzer modules.

While *registering new operators* the administrator enters appropriate data (and a photo if available) to the system database and programs his personal ID card.

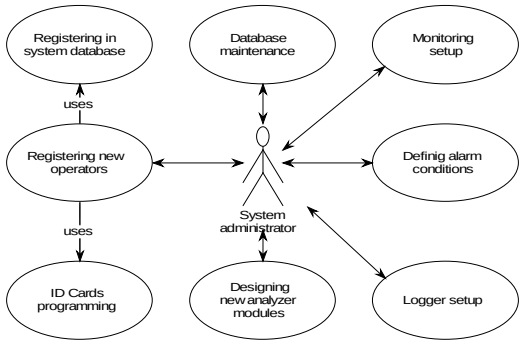


Figure 4. Administrative

Defining alarm conditions – the function enables setting up user-defined alarm conditions by writing condition-action rules (e.g. low saccadic activity during a longer period of time → inform operator's co-workers, wake him up using the beeper or playing appropriate sound and log the event in the database).

Designing new analyzer modules – based on earlier recorded data the administrator can create new analyzer modules that can recognize other behaviors than those which are built-in the system. The new modules are created using decision tree induction algorithm. The administrator names the new behavior to be recognized and points the data associated with it. The results received from the new modules can be used in alarm conditions.

Monitoring setup enables the administrator to choose the parameters to monitor as well as the algorithms of the desired accuracy to compute parameter values.

Logger setup provides tools for selecting the parameters to be recorded. For audio data sampling frequency can be chosen. As regards the video signal, a delay between storing consecutive frames can be set (e.g. one picture every two seconds).

Database maintenance – here the administrator can remove old or "uninteresting" data from the database. The "uninteresting" data is suggested by the built-in reasoning system.

3.2. Data Acquisition Unit (DAU)

In this section we describe the hardware part of the **BlueEyes** system with regard to the physiological data sensor, the DAU hardware components and microcontroller software.

3.2.1. Physiological data sensor

To provide the Data Acquisition Unit with necessary physiological data we decided to purchase an off-shelf eye movement sensor – Jazz Multisensor. It supplies raw digital data regarding eye position, the level of blood oxygenation, acceleration along horizontal and vertical axes and ambient light intensity.

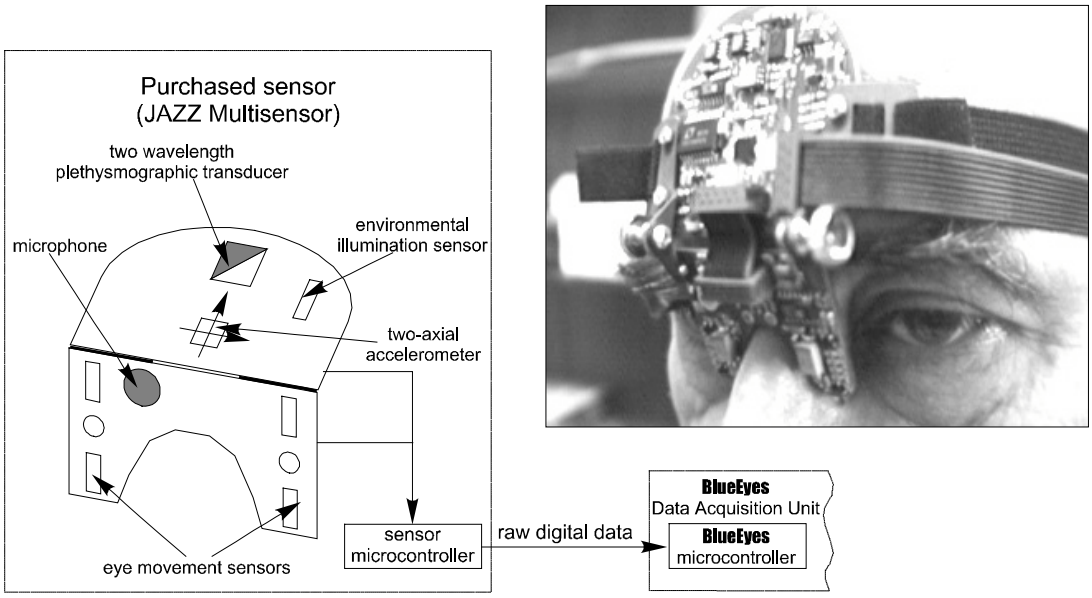


Figure 5. Jazz Multisensor

Eye movement is measured using direct infrared oculographic transducers.

The eye movement is sampled at 1kHz, the other parameters at 250 Hz. The sensor sends approximately 5,2kB of data per second.

3.2.2. Hardware specification

We have chosen Atmel 8952 microcontroller to be the core of the Data Acquisition Unit since it is a well-established industrial standard and provides necessary functionality (i.e. high speed serial port) at a low price. The figure below shows the other DAU components.

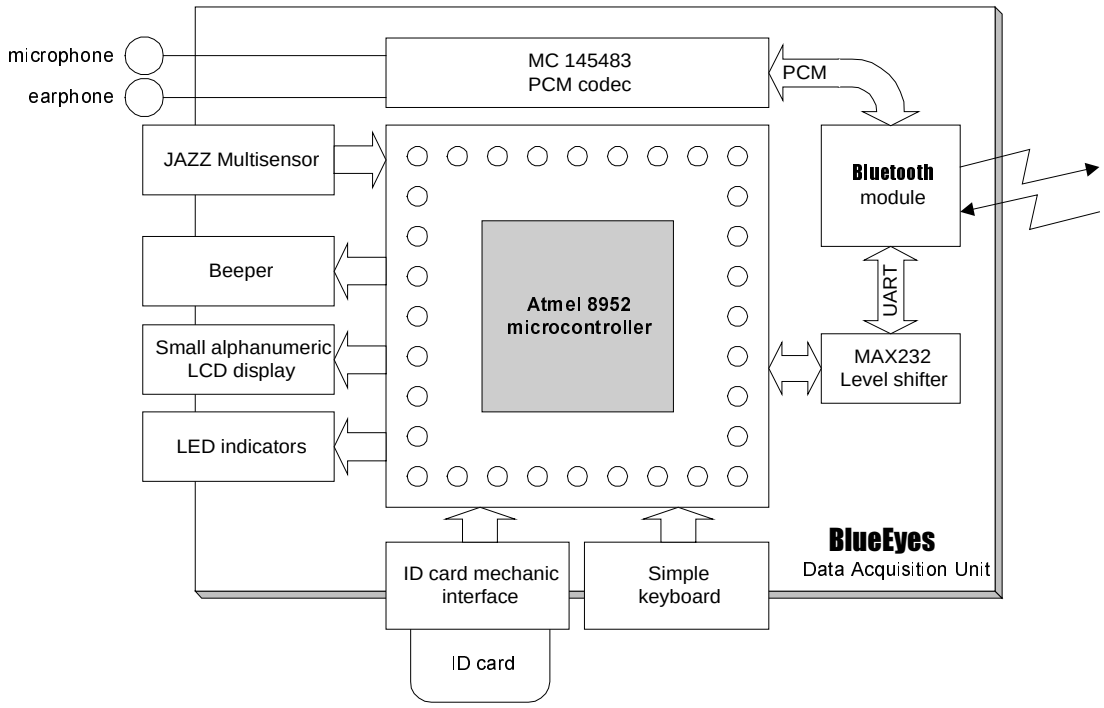


Figure 6. DAU hardware diagram

Since the Bluetooth module we received supports synchronous voice data transmission (SCO link) we decided to use hardware *PCM codec* to transmit operator's voice and central system sound feedback. The codec that we have employed reduces the microcontroller's tasks and lessens the amount of data being sent over the UART. Additionally, the Bluetooth module performs voice data compression, which results in smaller bandwidth utilization and better sound quality. We have chosen Motorola MC145483 linear 13-bit 3.3V codec to be used in the prototype, since it is voltage-level compatible with the Bluetooth module.

Communication between the Bluetooth module and the microcontroller is carried on using standard UART interface. *MAX232 Level Shifter* does the RS232 ↔ TTL voltage level conversion. The speed of the UART is set to 115200 bps in order to assure that the entire

sensor data is delivered in time to the central system (sensor data along with Bluetooth-related control data take up approx. 70% of the bandwidth).

The *alphanumeric LCD display* (two 8-character lines) gives more information of incoming events and helps the operator enter PIN code. In the prototype we used inexpensive Hitachi command-set LCD display.

The *LED indicators* show the results of built-in self-test, power level and the state of wireless connection.

The *simple keyboard* is used to react to incoming events (e.g. to silence the alarm sound) and to enter PIN code while performing authorization procedure.

ID card interface helps connect the operator's personal identification card to the DAU. After inserting the card authorization procedure starts. In the prototype as the *ID card* an I^2C EEPROM (ST14C16) was used since it provides easy programming interface and is capable of holding all the necessary operator's data. In the commercial release a cryptographic processor should be used instead. Each ID card is programmed to contain: operator's unique identifier, device access PIN code the operator enters on inserting his ID card and system access PIN code that is used on connection authentication. The operator's unique identifier enables the supervising system to distinguish different operators.

3.2.3. Microcontroller software specification

All the DAU software is written in 8051 assembler code, which assures the highest program efficiency and the lowest resource utilization. During the development we used GNU licensed 8051 assembler compiler – as31.

At a low-level design stage we modeled the software using a state diagram. Such a diagram facilitates implementation, debugging and testing phases.

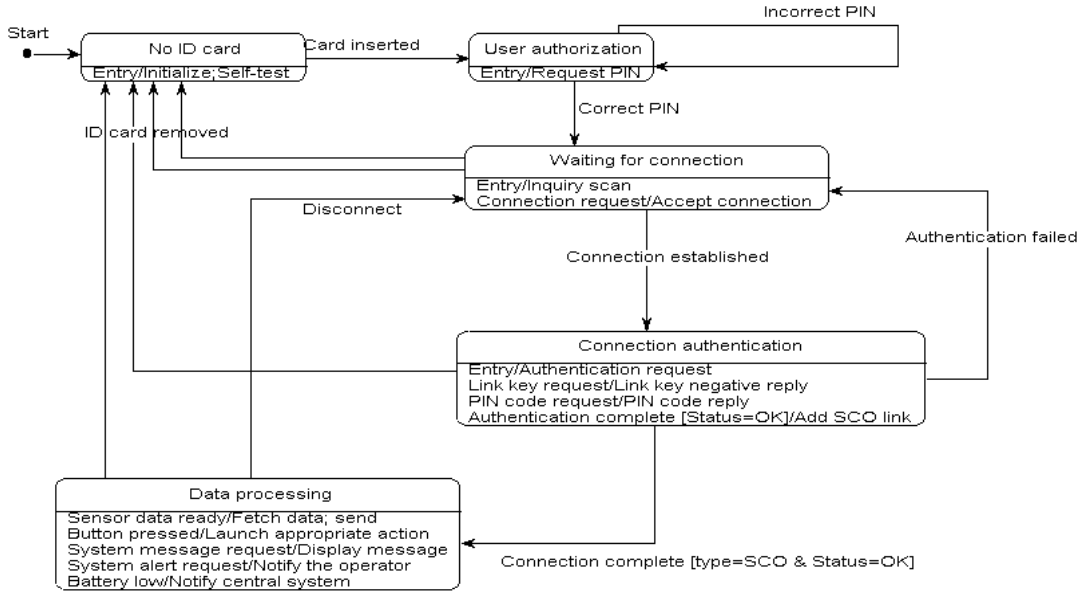


Figure 7. DAU state diagram

In each state the DAU communicates with the Bluetooth module using Host Controller Interface (HCI) commands.

In the *No ID card* state a self-test is performed to check if the device is working correctly. After the self-test passes the sensor and Bluetooth module are reset and some initialization commands are issued (i.e. HCI_Reset, HCI_Ericsson_Set_UART_Baud_Rate etc.). Once the initialization has been successfully completed the device starts to check periodically for ID card presence by attempting to perform an I^2C start condition. When the attempt succeeds and the operator's identifier is read correctly the device enters *User authorization* state.

In the *User authorization* state the operator is prompted to enter his secret PIN code. If the code matches the code read from the inserted ID card the device proceeds waiting for incoming Bluetooth connections.

On entering *Waiting for connection* state the DAU puts the Bluetooth module in Inquiry and Page Scan mode. After the first connection request appears, the DAU accepts it and enters *Connection authentication* state.

In the *Connection authentication* state the DAU issues Authentication_Requested HCI command. On Link Controller's Link_Key_Request the DAU sends Link_Key_Negative_Reply in order to force the Bluetooth module to generate the link key based on the supplied system access PIN code. After a successful authentication the DAU enters the *Data Processing* state, otherwise it terminates the connection and enters the *Waiting for connection* state.

The main DAU operation takes place in the *Data Processing* state. In the state five main kinds of events are handled. Since the sensor data has to be delivered on time to the central system, data fetching is performed in high-priority interrupt handler procedure. Every 4ms the Jazz sensor raises the interrupt signaling the data is ready for reading. The following data frame is used:

24 bits	12 b	12 b	12 b	12 b	12 b	12 b	24 bits	12 b	12 b	12 b	12 b	12 b	12 b
Preamble	EyeX	EyeY	AccX	AccY	EyeX	EyeY	PulsoOxy	EyeX	EyeY	Batt	Light	EyeX	EyeY

Figure 8. Jazz sensor frame format

The preamble is used to synchronize the beginning of the frame, EyeX represents the horizontal position of the eye, EyeY – vertical, AccX and AccY – the acceleration vectors along X and Y axes, PulsoOxy, Batt and Light – blood oxygenation, voltage level and light intensity respectively. The figure below shows the sensor communication timing.

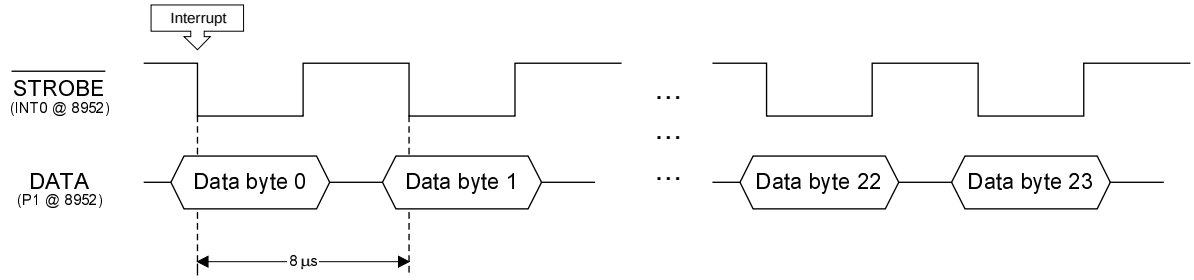


Figure 9. Jazz Sensor data fetching waveforms

The received data is stored in an internal buffer; after the whole frame is completed the DAU encapsulates the data in an ACL frame and sends it over the Bluetooth link. The fetching phase takes up approx. 192μs (24 frames x 8μs) and the sending phase takes at 115200 bps approx. 2,8 ms, so the timing fits well in the 4ms window. Omitting the acknowledgement signal allows faster transmission.

The second group of events handled in the *Data Processing* state are system messages and alerts. They are sent from the central system using the Bluetooth link. Since the communication also uses microcontroller's interrupt system the events are delivered instantly. More detailed considerations regarding handling of the Bluetooth- and sensor-related interrupts are presented in section 3.2.4 Tradeoffs and optimizations.

The remaining time of the microcontroller is utilized performing LCD display, checking the state of the buttons, ID card presence and battery voltage level. Depending on which button is pressed appropriate actions are launched (e.g. dismissing alarm sound). In case the voltage level is low the operator and the central system are notified.

In every state removing the ID card causes the device to enter the *No ID card* state, terminating all the established connections.

3.2.4. DAU Project and implementation tradeoffs and optimizations

In the DAU there are two independent data sources – the Jazz sensor and Bluetooth Host Controller. Since they are both handled using the interrupt system it is necessary to decide which of the sources should have higher priority. Giving the sensor data the highest priority may result in losing some of the data sent by the Bluetooth module, as the transmission of the sensor data (192 μs) takes twice as much time as receiving one byte from UART (87 μs). Missing one single byte sent from the Bluetooth causes the loss of control over the transmission. On the other hand, giving the Bluetooth the highest priority will make the DAU

stop receiving the sensor data until the Host Controller finishes its transmission. In this case the interrupted sensor frame shall be discarded. We do not consider the data being sent *from* the DAU to the Bluetooth as it does not affect the operation. Since missing one byte of the Bluetooth communication affects functioning of the DAU much more than losing one single sensor data frame (which can be estimated at the central system) we have chosen the second option. Central system alerts are the only signals that can appear during sensor data fetching after all the unimportant Bluetooth events have been masked out. The best solution would be to make the central unit synchronize the alerts to be sent with the Bluetooth data reception. As the delivered operating system is not a real-time system, the full synchronization is not possible. We therefore have calculated the probability of interrupting the sensor data transmission by a single alert.

The interruption may occur when there is a collision between sensor data fetching and an asynchronously incoming central system alert. The situation is shown in the figure below.

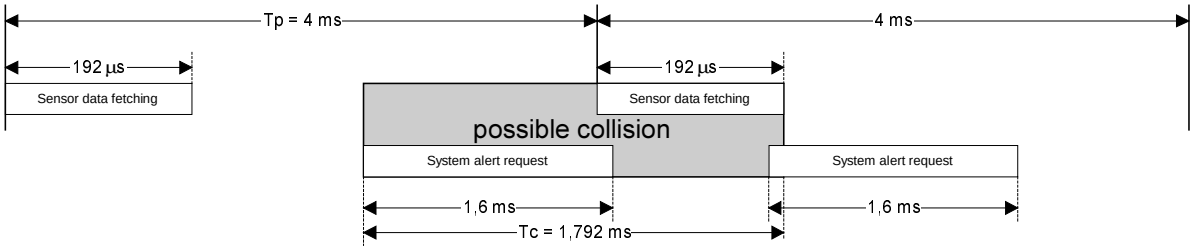


Figure 10. Collision condition timing

The alert message is approx. 20 bytes long (including HCI headers), thus its reception time is approx. 1.6 ms. The probability of the collision P_c equals T_c/T_p = 1.792ms / 4ms = 0.44, which means every second alert request discards one sensor data frame. Assuming that during normal system operation the alert message arrivals are less frequent than one per 15 minutes – only one of more than 200,000 sensor data frames is lost. Moreover, while buffering at the central system the missing frame is reconstructed based on the previous and next sensor data packet (each frame's preamble contains a sequential number and a checksum).

As the Bluetooth module communicates asynchronously with the microcontroller there was a need of implementing a cyclic serial port buffer, featuring UART CTS/RTS flow control and a producer-consumer synchronization mechanism.

At the low-level design stage we calculated the amount of data to be sent to the Bluetooth device over UART interface. The sensor data along with Bluetooth control command occupy approx. 70% of the available 115200 bps bandwidth. Thus, if we wanted to send the operator's voice data over the same channel this would dramatically increase microcontroller's load and affect sound quality. Introducing hardware PCM codec and using the Bluetooth synchronous connection (SCO link) eliminates those constraints.

During the DAU hardware design and implementation we put much emphasis on optimizing price and power consumption of the device. To achieve this we used standard low-voltage ICs where possible. Moreover, the microcontroller software enables power-down modes of the ICs when it does not disturb the function of the device (i.e. when no ID card is inserted the PCM codec and the Bluetooth module are put in low-power mode). All those optimizations help keep the time between recharging the batteries long.

3.2.5. DAU Validation and testing

While testing the DAU software we used two techniques. At the implementation stage we were performing a white-box testing. After completing subsequent portions of an assembler code its creator checked the program for hidden or logical errors. Then, another team member checked the results. After the implementation phase finished, based on the state diagram we performed a black-box testing. All the transitions between the states have been checked against possible implementation errors. This assured us that every state is reached only under the conditions specified on the diagram. After the implementation of the central system finished we performed integrated system tests.

3.3. Central System Unit (CSU)

CSU software is located on the delivered Toshiba laptop, in case of larger resource demands the processing can be distributed among a number of nodes. In this section we describe the four main CSU modules (see Fig. 1): Connection Manager, Data Analysis, Data Logger and Visualization. The modules exchange data using specially designed single-producer-multi-consumer buffered thread-safe queues. Any number of consumer modules can register to receive the data supplied by a producer. Every single consumer can register at any number of producers, receiving therefore different types of data. Naturally, every consumer may be a producer for other consumers. This approach enables high system scalability – new data processing modules (i.e. filters, data analyzers and loggers) can be easily added by simply registering as a consumer.

3.3.1. Connection Manager

Connection Manager's main task is to perform low-level Bluetooth communication using Host Controller Interface commands. It is designed to cooperate with all available Bluetooth

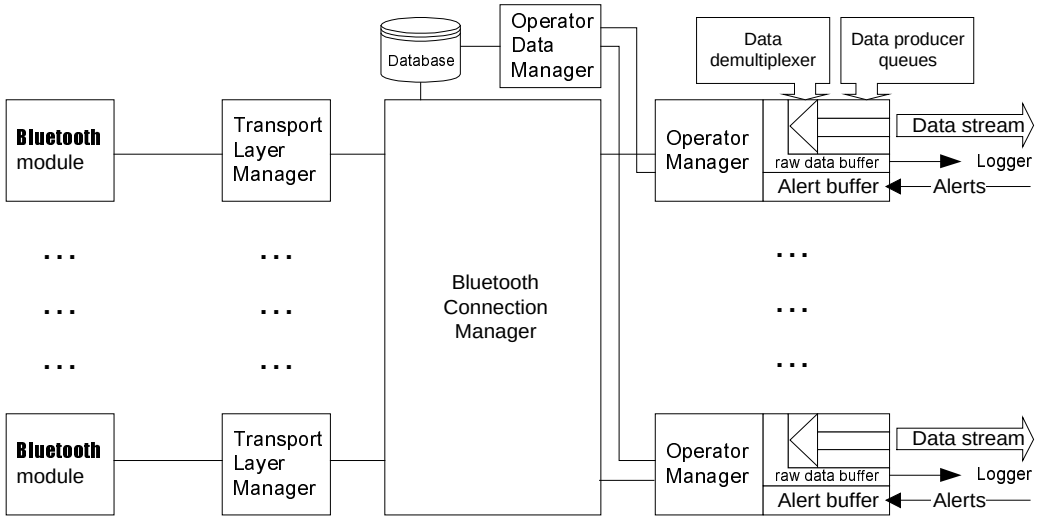


Figure 11. Connection Manager components

devices in order to support roaming. Additionally, Connection Manager authorizes operators, manages their sessions, demultiplexes and buffers raw physiological data. Figure 11 shows Connection Manager architecture.

Transport Layer Manager hides the details regarding actual Bluetooth physical transport interface (which can be either RS232 or UART or USB standard) and provides uniform HCI command interface.

Bluetooth Connection Manager is responsible for establishing and maintaining connections using all available Bluetooth devices. It periodically inquires new devices in an operating range and checks whether they are registered in the system database. Only with those devices the Connection Manager will communicate. After establishing a connection an authentication procedure occurs. The authentication process is performed using system PIN code fetched from the database. Once the connection has been authenticated the mobile unit sends a data frame containing the operator's identifier. Finally, the Connection Manager adds a SCO link (voice connection) and runs a new dedicated Operator Manager, which will manage the new operator's session. Additionally, the Connection Manager maps the operator's identifiers into the Bluetooth connections, so that when the operators roam around the covered area a connection with an appropriate Bluetooth device is established and the data stream is redirected accordingly.

The data of each supervised operator is buffered separately in the dedicated *Operator Manager*. At the startup it communicates with the Operator Data Manager in order to get more detailed personal data. The most important Operator Manager's task is to buffer the incoming raw data and to split it into separate data streams related to each of the measured parameters. The raw data is sent to a Logger Module, the split data streams are available for the other system modules through producer-consumer queues. Furthermore, the Operator Manager provides an interface for sending alert messages to the related operator.

Operator Data Manager provides an interface to the operator database enabling the other modules to read or write personal data and system access information.

3.3.2. Data Analysis Module

The module performs the analysis of the raw sensor data in order to obtain information about the operator's physiological condition. The separately running Data Analysis Module supervises each of the working operators. The module consists of a number of smaller analyzers extracting different types of information. Each of the analyzers registers at the appropriate Operator Manager or another analyzer as a data consumer and, acting as a producer, provides the results of the analysis. An analyzer can be either a simple signal filter (e.g. Finite Input Response (FIR) filter) or a generic data extractor (e.g. signal variance, saccade detector) or a custom detector module. As we are not able to predict all the supervisors' needs, the custom modules are created by applying a supervised machine learning algorithm to a set of earlier recorded examples containing the characteristic features to be recognized. In the prototype we used an improved C4.5 decision tree induction algorithm. The computed features can be e.g. the operator's position (standing, walking and lying) or whether his eyes are closed or opened.

As built-in analyzer modules we implemented a saccade detector, visual attention level, blood oxygenation and pulse rate analyzers.

The *saccade detector* registers as an eye movement and accelerometer signal variance data consumer and uses the data to signal saccade occurrence. Since saccades are the fastest eye movements the algorithm calculates eye movement velocity and checks physiological constraints.

The algorithm we used performs two main steps:

A. *User adjustment step*. The phase takes up 5 s. After buffering approx. 5 s of the signal differentiate it using three point central difference algorithm, which will give eye velocity time series. Sort the velocities by absolute value and calculate upper 15% of the border velocity along both X – v_{0x} and Y – v_{0y} axes . As a result v_{0x} and v_{0y} are cut-off velocities.

B. *On-line analyzer flow*. Continuously calculate eye movement velocity using three point central difference algorithm. If the velocity excess precalculated v_0 (both axes are considered separately) there is a possibility of saccade occurrence. Check the following conditions (if any of them is satisfied do not detect a saccade):

- the last saccade detection was less than 130 ms ago (physiological constraint – the saccades will not occur more frequently)
- the movement is nonlinear (physiological constraint)
- compare the signal with accelerometer (rapid head movement may force eye activity of comparable speed)
- if the accelerometer signal is enormously uneven consider ignoring the signal due to possible sensor device movements.

If none of the above conditions is satisfied – signal the saccade occurrence.

The *visual attention level analyzer* uses as input the results produced by the saccade detector. Low saccadic activity (large delays between subsequent saccades) suggests lowered visual attention level (e.g. caused by thoughtfulness). Thus, we propose a simple algorithm that calculates the visual attention level (L_{va}): $L_{va} = 100/t_{s10}$, where t_{s10} denotes the time (in seconds) occupied by the last ten saccades. Scientific research has proven [1] that during normal visual information intake the time between consecutive saccades should vary from 180 up to 350 ms. This gives L_{va} at 28 up to 58 units. The values of L_{va} lower than 25 for a longer period of time should cause a warning condition. The following figure shows the situation where the visual attention lowers for a few seconds.

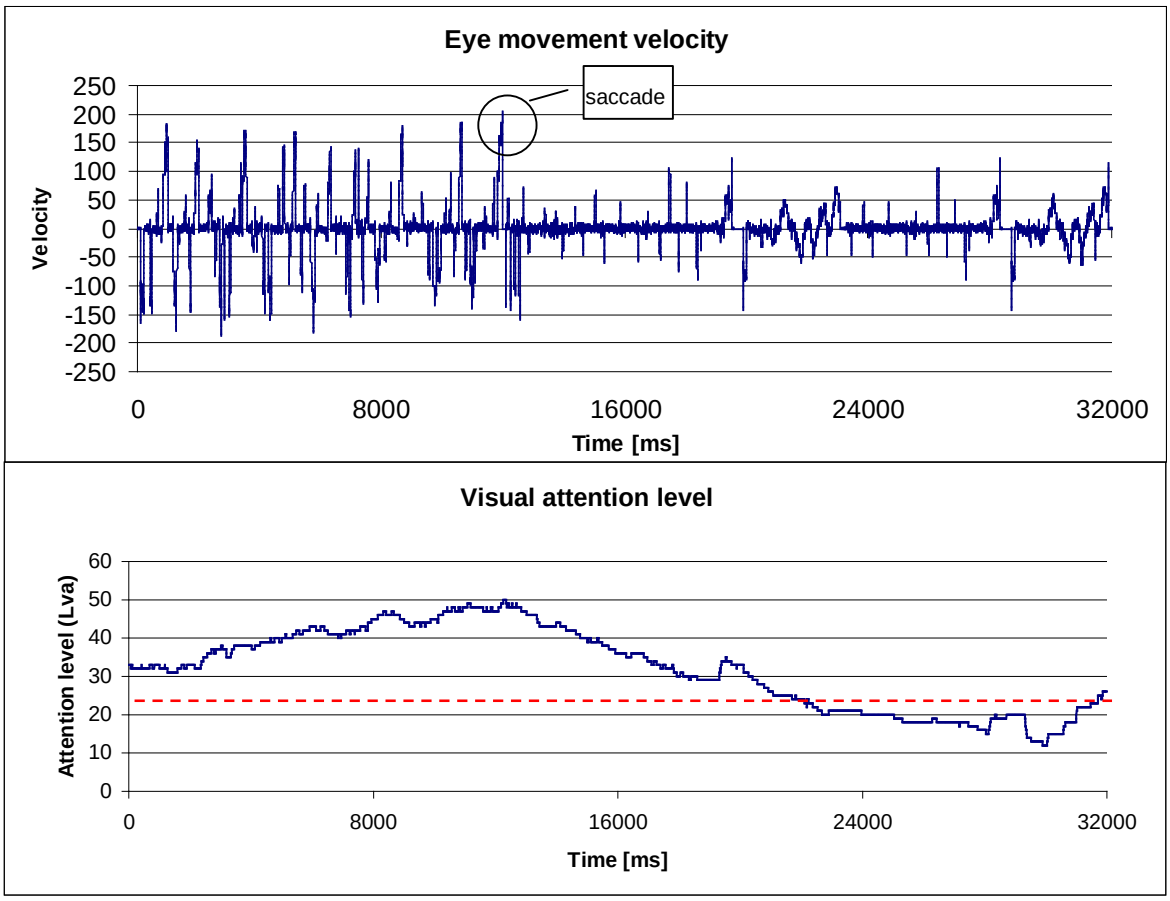


Figure 12. Saccade occurrences and visual attention level

The *Pulse rate analyzer* registers for the oxyhemoglobin and deoxyhemoglobin level data streams. Since both signals contain a strong sinusoidal component related to heartbeat, the pulse rate can be calculated measuring the time delay between subsequent extremes of one of the signals. We decided not to process only one of the data streams – the algorithm is designed to choose dynamically one of them on the grounds of the signal level. Unfortunately, the both signals are noised so they must be filtered before further processing. We considered a number of different algorithms and decided to implement average value based smoothing. More detailed discussion is presented in section 3.3.5 Tradeoffs and Optimization. The algorithm consists in calculating an average signal value in a window of 100 samples. In every step the window is advanced 5 samples in order to reduce CPU load. This approach lowers the sampling rate from 250 Hz down to 50 Hz. However, since the

heartbeat frequency is at most 4 Hz the Nyquist condition remains satisfied. The figures show the signal before (Fig. 13a) and after filtering (Fig 13b).

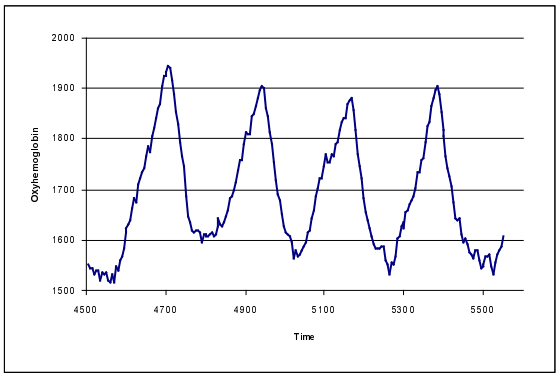


Figure 13a.

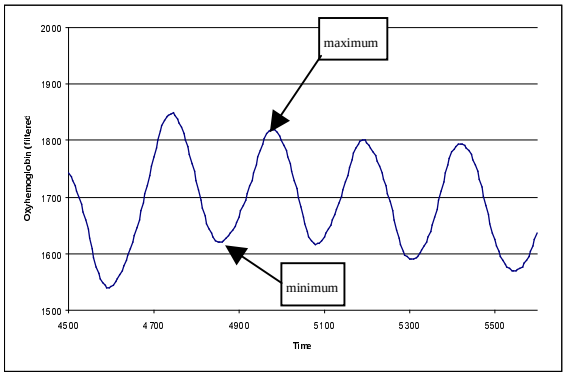


Figure 13b.

After filtering the signal the pulse calculation algorithm is applied. The algorithm chooses the point to be the next maximum if it satisfies three conditions: points on the left and on the right have lower values, the previous extreme was a minimum, and the time between the maximums is not too short (physiological constraint). The new pulse value is calculated based on the distance between the new and the previous maximum detected. The algorithm gets the last 5 calculated pulse values and discards 2 extreme values to average the rest. Finally, it does the same with the minimums of the signal to obtain the second pulse rate value, which gives the final result after averaging.

Additionally, we implemented a simple module that calculates average blood oxygenation level. Despite its simplicity the parameter is an important measure of the operator's physiological condition.

The other signal features that are not recognized by the built-in analyzers can be extracted using custom modules created by *Decision Tree Induction* module. The custom module processes the generated decision tree, registers for needed data streams and produces the desired output signal.

Decision Tree Induction module generates the decision trees, which are binary trees with an attribute test in each node. The decision tree input data is an object described by means of a set of attribute-value pairs. The algorithm is not able to process time series directly. The attributes therefore are average signal value, signal variance and the strongest sinusoidal components. As an output the decision tree returns the category the object belongs to.

In the *Decision Tree Induction* module we mainly use C 4.5 algorithm [2], but also propose our own modifications. The algorithm is a supervised learning from examples i.e. it considers both attributes that describe the case and a correct answer. The main idea is to use a divide-and-conquer approach to split the initial set of examples into subsets using a simple rule (i-th attribute less than a value). Each division is based on entropy calculation – the distribution with the lowest entropy is chosen. Additionally, we propose many modifications concerning some steps of the algorithm and further exploration of the system.

Algorithm step	Original C 4.5	Proposed modification
Entropy calculation	$E(S)=-\sum_{j=1}^{k}\frac{f(C_i,S)}{\lvert S\rvert}*\log\left(\frac{f(C_i,S)}{\lvert S\rvert}\right)$ where $f(x,y)$ denotes the frequency of decision class x in set y	Decision not based on entropy. "Common sense" rules for splitting input set.
Choosing best division	Minimization of weighted entropy sum of subsets	Additional factor that markedly favors balanced divisions
Division condition	Condition attribute $a_i < v$	Additional condition attribute $a_i \in (a,b)$
Tree pruning	Prune the tree to maximize the test set hit ratio	Stop splitting input set when it is too small (much faster than the other method). No tree modifications after testing.

For each case to be classified C 4.5 traverses the tree until reaching the leaf where appropriate category id is stored. To increase the hit ratio our system uses more advanced procedure. For single analysis we develop a group of k trees (where k is a parameter), which

we call a decision forest. Initial example set S is divided randomly into k+1 subsets $S_0,...,S_k$. S_0 is left to test the whole decision forest. Each tree is induced using various modifications of the algorithm to provide results' independence. Each i-th tree is taught using $S\backslash S_0\backslash S_i$ set (S without S_0 and S_i sets) and tested with S_i that estimates a single tree error ratio. Furthermore we extract all wrongly classified examples and calculate correlation matrix between each pair of the trees. In an exploring phase we use unequal voting rule – each tree has a vote of strength of its reliability. Additionally, if two trees give the same answer their vote is weakened by the correlation ratio.

Alarm Dispatcher Module is a very important part of the Data Analysis module. It registers for the results of the data analysis, checks them with regard to the user-defined alarm conditions and launches appropriate actions when needed. The module is a producer of the alarm messages, so that they are accessible in the logger and visualization modules.

3.3.3. Data Logger Module

The module provides support for storing the monitored data in order to enable the supervisor to reconstruct and analyze the course of the operator's duty. The module registers as a consumer of the data to be stored in the database. Each working operator's data is recorded by a separate instance of the Data Logger. Apart from the raw or processed physiological data, alerts and operator's voice are stored. The raw data is supplied by the related Operator Manager module (Fig. 11), whereas the Data Analysis module delivers the processed data. The voice data is delivered by a Voice Data Acquisition module. The module registers as an operator's voice data consumer and optionally processes the sound to be stored (i.e. reduces noise or removes the fragments when the operator does not speak). The Logger's task is to add appropriate time stamps to enable the system to reconstruct the voice.

Additionally, there is a dedicated video data logger, which records the data supplied by the Video Data Acquisition module (in the prototype we use JPEG compression). The module is designed to handle one or more cameras using Video For Windows standard. The Data Logger is able to use any ODBC-compliant database system. In the prototype we used MS SQL Server, which is a part of the Project Kit.

3.3.4. Visualization Module

The module provides user interface for the supervisors. It enables them to watch each of the working operator's physiological condition along with a preview of selected video source and his related sound stream. All the incoming alarm messages are instantly signaled to the supervisor. Moreover, the visualization module can be set in the off-line mode, where all the data is fetched from the database. Watching all the recorded physiological parameters, alarms, video and audio data the supervisor is able to reconstruct the course of the selected operator's duty.

3.3.5. CSU Project and implementation tradeoffs and optimizations

The prototype we have built has several limitations, which are not the result of the project deficiency but are rather caused by the constraints imposed by the Project Kit and small budget. In the commercial release the USB web-cam should be replaced by an industrial camera, connected to a capturing device (optionally a video signal multiplexer could be used). The use of such a camera would lessen CPU load and improve the video signal quality. Since the Bluetooth module we received does not support redirecting of the voice SCO connections' data to the serial port (which is a part of the Bluetooth specification) we had to build a PCM interface on the central system side similar to the one used in the DAU. This makes the Voice Data Acquisition module receive the sound using the sound card.

Implementing the pulse rate analyzer we considered several filtering algorithms. We tested FIR (Finite Impulse Response) digital filters, whose output signal is easier to analyze. However, the filter needs ten times as many samples as our algorithm to produce satisfactory results. It makes the FIR filter consume much more CPU processing power. The use of FFT algorithm is pointless, as it needs over 20 000 samples to achieve the desired accuracy (i.e. 0.01 Hz).

While designing and implementing the reasoning algorithm, apart from the decision trees, we considered neural networks and k-nn (k-nearest neighbors) algorithm. We have chosen the decision trees, as the other algorithms are not able to explain their decisions. Traversing the tree nodes it is possible to identify all the perquisites of the decision, whereas neural network and k-nn coefficients do not have clear interpretation. Additionally, based on the decision tree structure the system is able to generate equivalent decision rule set.

3.3.6. CSU Validation and testing

While implementing and testing particular central system modules we used about a hundred disk files containing prerecorded sensor data. The data was specially selected to contain the behaviors we were interested in (e.g. saccades, breathing, low and high pulse rate, standing, walking, lying). The files were prepared using a dedicated tool (section 3.4). As all the central system modules communicate using producer-consumer queues it was possible to implement and test the modules independently. In order to achieve it we created temporary producers serving the data read from the selected file. The next testing step was to create a producer reading the data directly from the sensor using a parallel port. This enabled us to test the Data Analysis, Logger and Visualization modules working in a cooperation with a real-time data source. At this stage we generated a custom analyzer module which determines the position of the operator (standing, lying, walking). After testing we found out that the

generated module properly classifies the position. We also did some minor algorithm and database access optimizations. In the integrated system testing phase we added the wireless communication facility activating the DAU and Connection Manager modules. The last task was to check the Bluetooth communication performance with regard to the number of lost data frames.

3.4. Tools developed during the course of the project

In creating the hardware part of the DAU we built a development board, which enabled us to mount, connect and test various peripheral devices cooperating with the microcontroller.

During the implementation of the DAU we needed a piece of software to establish and test Bluetooth connections. We therefore created a tool called **BlueDentist** (Fig. 14). The tool provides support for controlling the currently connected Bluetooth device. Its functions are: local device management (resetting, reading local BD_ADDR, putting in Inquiry/Page and Inquiry/Page scan modes, reading the list of locally supported features and setting UART speed) and connection management (receiving and displaying Inquiry scan results, establishing ACL links, adding SCO connections, performing link authorization procedure, sending test data packets and disconnecting).

To test the possibilities and performance of the remaining parts of the Project Kit (computer, camera and database software) we created **BlueCapture** (Fig. 15). The tool supports capturing video data from various sources (USB web-cam, industrial camera) and storing the data in the MS SQL Server database. Additionally, the application performs sound recording. After filtering and removing insignificant fragments (i.e. silence) the audio data is stored in the database. Finally, the program plays the recorded audiovisual stream. We used the software to measure database system performance and to optimize some of the SQL queries (e.g. we replaced correlated SQL queries with cursor operations). Since all the components of the

application have been tested thoroughly they were reused in the final software, which additionally reduced testing time.

We also created a simple tool for recording Jazz Multisensor measurements. The program reads the data using a parallel port and writes it to a file.

To program the operator's personal ID card we use a standard parallel port, as the EEPROMs and the port are both TTL-compliant. A simple dialog-based application helps to accomplish the task.

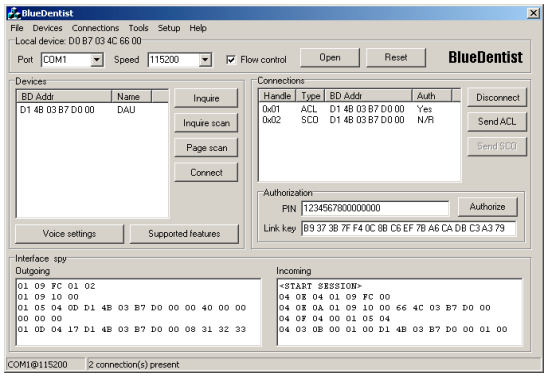


Figure 14. BlueDentist

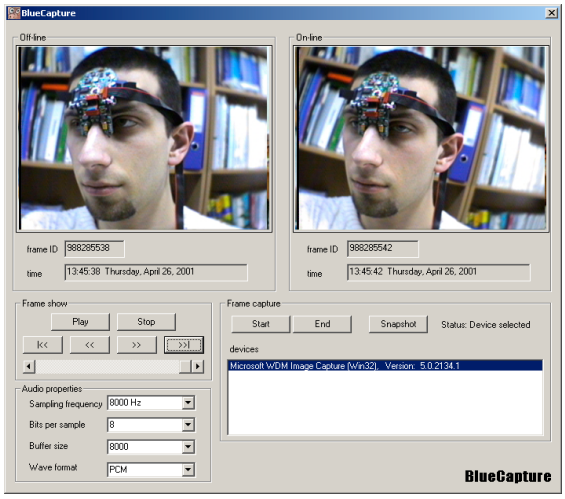


Figure 15. BlueCapture

3.5. Additional costs

Item	Quantity	Unit Price	Cost
Jazz Multisensor (OEM version)	1	$130.00	$130.00
AT89c52 microcontroller	1	$4.50	$4.50
MC145483 PCM codec	2	$3.30	$6.60
PCM clock generator	2	$3.20	$6.40
HD44780A00 LCD display	1	$7.10	$7.10
MAX232	1	$1.10	$1.10
Earphone	1	$1.30	$1.30
ST14c16 EEPROM	1	$2.30	$2.30
ID card interface	1	$4.90	$4.90
Passive elements			$10,00
Plugs, sockets, case, bards			$15.00
		TOTAL	$189.20

4. Summary

We have decided to develop the BlueEyes system because of the need for a real-time monitoring system for a human operator. The approach is innovative since it helps supervise the operator not the process, as it is in presently available solutions. We hope the system in its commercial release will help avoid potential threats resulting from human errors, such as weariness, oversight, tiredness or temporal indisposition. However, the prototype we have developed is a good estimation of the possibilities of the final product.

We find it possible still to improve our project. The use of a miniature CMOS camera integrated into the eye movement sensor will enable the system to calculate the point of gaze and observe what the operator is actually looking at. Introducing voice recognition algorithm will facilitate the communication between the operator and the central system and simplify authorization process.

Despite considering in the report only the operators working in control rooms, our solution may well be applied to everyday life situations. Assuming the operator is a driver and the supervised process is car driving it is possible to build a simpler embedded on-line system, which will only monitor conscious brain involvement and warn when necessary. As in this case the logging module is redundant, and the Bluetooth technology is becoming more and more popular, the commercial implementation of such a system would be relatively inexpensive.

The final thing is to explain the name of our system. BlueEyes emphasizes the foundations of the project – **Blue**tooth technology and the movements of the **eyes**. Bluetooth provides reliable wireless communication whereas the eye movements enable us to obtain a lot of interesting and important information.

5. References

1. Carpenter R. H. S., Movements of the eyes, 2nd edition, Pion Limited, 1988, London.
2. Quinlan J. R., C4.5 programs for machine learning, Morgan Kaufmann Publishers, 1993, San Mateo, California.
3. Horowitz P., Hill W., The art of Electronics, Cambridge University Press, 1989.
4. Bluetooth specification, version 1.0B, Bluetooth SIG, 1999.
5. ROK 101 007 Bluetooth Module, Ericsson Microelectronics, 2000.
6. MC145483 3V 13-bit Linear PCM Codec-Filter, Motorola Semiconductor, 1997.
7. AT89C52 8-bit Microcontroller Datasheet, Atmel.
8. ST14C16 memory card IC Datasheet, ST Microelectronics, 1999.
9. Leinecker R. C., Archer T., Visual C++ Bible, IDG Books, 1999.
10. MSDN Library, Microsoft (Visual Studio 6.0).
11. Intel Signal Processing Library – Reference Manual.