**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

The ability of computers to enhance and augment both mental and physical abilities and potential is no longer the exclusive realm of science fiction writers. It is becoming a reality. A brain-computer interface (BCI) is a new communication channel between the human brain and a digital computer, i.e., collaboration between a brain and a device that enables signals from the brain to direct some external activity, such as control of a cursor or a prosthetic limb. It isn't about convenience for severely disabled people, development of a brain-computer interface (BCI) could be the most important technological breakthrough in decades. This system not only is limited to usage of patients with a particular disease but maybe used for a wide variety of patients with disease which include Amyotrophic Lateral Sclerosis, Brachial plexus injury, Brain injury, Cerebral Palsy, Friedrich’s Ataxia, Guillain-Barre Syndrome, Multiple Sclerosis, Muscular Dystrophy, Post-Polio Syndrome, Spina Bifida, Spinal Cord Injury, Stroke, Syringomyelia/Tethered cord, Transverse Myelitis.

Many people imagine that BCI will allow them to simply think of a word or phrase and have it appear on the screen, or control a wheelchair by thinking about where they want to go. Unfortunately, this is not the case with current BCI technology. There are a variety of BCI systems, and each one works a little differently. The very reason a BCI works is because of the way our brain functions. Our brains are filled with neurons, individual nerve cells connected to one another by dendrites and axons. Every time we think, move, feel or remember something, our neurons are at work. That work is carried out by small electric signals that zip from neuron to neuron as fast as 250 miles per hour. The signals are generated by differences in electric potential carried by ions on the membrane of each neuron. By reading signals from an array of neurons and using computer chips and programs to translate the signals into action, BCI can enable a person suffering from paralysis to write a book or control a motorized wheelchair or prosthetic limb through thought alone. Current brain-interface devices require deliberate conscious thought; some future applications, such as prosthetic control, are likely to work effortlessly. Since this technology is being developed for a much larger market than those with neurological disease, in the future we will see BCI devices available for many computer functions.

**1.2 PROBLEM DESCRIPTION**

P300 algorithms are based only on attentiveness of a person which accounts for binary control only. Though wide channel BCIs are available, it is either too costly (which can neither be mass produced nor mass distributed) or it becomes a nascence to attach/operate. Traditional BCI instruments require wet electrodes which are hard to attach and calibrate. Considering this a factor, current BCI systems these days use dry electrodes in order to operation which only tends to add more to the instrument cost.

In order to cut down on the cost, commercial BCI systems these days use a single channel for their execution. This in turn results in restricts the system to binary control. Another major problem caused by binary control is high time complexity. Most BCI spelling systems display a series of letters, either one at a time or by highlighting letters in a grid. When the letter you want lights up, your brain wave changes. The computer looks for that change and interprets it as a 'keystroke'. For example, if you wanted to type the letter A, you would focus on the A and count each time it flashed, or think “Yes!” when you see it appear on the screen. Recognizing the A would trigger a spike in your brain signals, which would be detected by the BCI system. Usually, each letter must be ‘selected’ multiple times, so typing with a BCI is quite slow. Systems designed to control a computer cursor often rely on movement imagery.

You would imagine squeezing your right hand to move the cursor to the right, and your left hand to move the cursor to the left. By this way you cannot directly use multiple directions of control like we use on many electronics hence it increases the time complexity multiple times for simple tasks which would otherwise take only take negligible amount of time otherwise. Any operation performed using the previous system is restricted to creation of textual strings. Any practical application of these types of systems is restricted to displaying of the created textual stings on the screen rather than execution of any sort. For a disabled person, this may not be a significant leap forward in control as another person is still needed at all times near the patient in order to perform any task required by the patient.

**1.3 OBJECTIVE**

The objectives of the proposed system are

* To develop a computer interaction system that can provide a communication interface for physically disabled people using EEG.
* To overcomes the problem of time complexity.
* To make the system executable i.e., to provide extended control to system rather than just textual string editing.

**1.4 SCOPE OF THE PROJECT**

The scope of the project extend to the following,

* Communication Pathway for physically disabled.
* Automation of control for electrical/electronic devices via Brainwaves.

**1.5 ORGANISATION OF THE REPORT**

The thesis of the project is organized as follows: The Chapter 2 surveys and brings out existing systems and research works in this thrust area namely Brain Computer Interface. It summarises the issues from the various BCI based research works that are being handled in this field. The Chapter 3 describes modular design and Functional architecture that are to be implemented. System architecture and the functional design of the newly proposed EEG based BCI system is expressed in this chapter. Chapter 4 describes the implementation details of each module and presents the input/output screenshots. It also gives us a detailed view of how each module appears to the user in terms of visual aspect. Various Testing at Unit, Module and Integrated levels are covered in Chapter 5.Performance analysis of system had also been brought out in Chapter 5. Chapter 6 contains the conclusion of the project and future enhancements that can be implemented to the proposed system.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 INTRODUCTION**

Literature survey is the documentation of a comprehensive review of the publishers and unpublished work from the secondary sources data in the areas of specific interest to the researcher. It is a survey of all the techniques used to retrieve data efficiently by deploying Two Round Searchable Encryption scheme.

**2.2 SURVEY OF THE EXISTING SYSTEMS**

In previous systems, infra-red rays and cameras were used to detect the movement of eyes for disabled people. But for these systems there were sever setbacks namely

1. For the camera based system, the image processor had to find the position of the eye inside the image frame first in order to process any user input from the user. This caused a delay in execution of the system and also slack sometimes.
2. For the infra-red based system, an infra-red beam sensor had to be placed close to the eye. Constant exposure of eye could cause ill-effects for the eye.
3. A person had to stare on the screen the entire time in order for the system to execute properly. This caused a nuisance sometimes as a person constantly staring cannot stare at the screen for long hours.

In order to overcome these problems, BCI (Brain Computer Interface) which used wet/dry electrodes measuring the brain waves. The following is a survey of the existing approaches.

**2.2.1 Visual and Auditory Brain–Computer Interfaces**

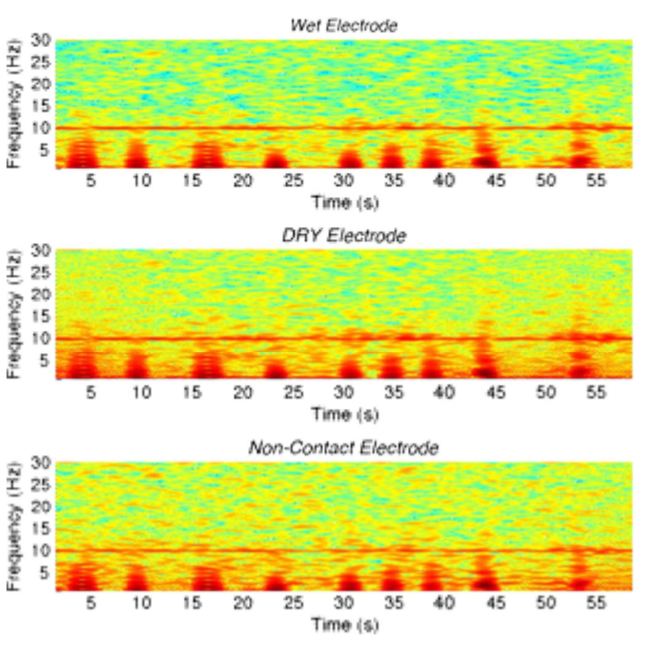
This paper focused on addressing the brain waves into categories from they can be harnessed for usage in various purposes. From the paper we can infer that each type of wave extracted from the brain can be used for various reasons. This paper focuses on two types of BCI namely visual (v-BCI) and auditory brain computing system (a-BCI) for which Exogenous Stimulus and Endogenously stimulus is used.

1. Exogenous Stimulus is further classified into
   * 1. Visual Evoked Potentials VEP
        1. Transient VEP (TVEP) under low-rate stimulus condition (<2 Hz).
        2. Steady-state VEP (SSVEP) under high-rate stimulus condition (>6 Hz).
        3. Motion VEP, which reﬂects visual motion processing.
        4. Code-modulated VEP, which can be elicited by a pseudo-random stimulus sequence.
     2. Auditory Steady-State Responses (ASSR).
2. Endogenously Modulated Brain Signals
   1. Response to oddball stimulus (auditory mismatch negativity (MMN), N200 and P300);
   2. Response to mental tasks [late positive components(LPC)]
   3. Response inhibition (No-Go N2)
   4. Semantic processing (N400)
   5. Attention-modulated brain signals (SSVEP, ASSR)

In previous study, the VEP-based BCI systems were categorized using this classiﬁcation method. This study further extends the taxonomy to classify all v-BCI and a-BCI systems in a comprehensive and systematic way. Similar to the classiﬁcation of the telecommunication systems, the v-BCI and a-BCI systems can be sorted into the following ﬁve groups: 1) TDMA, 2) FDMA, 3) CDMA, 4) SDMA and 5) HMA. In this way, v-BCI and a-BCI systems can be examined under a uniﬁed framework based on multiple target access methods. There are three primary advantages to this categorization. First, it simpliﬁes the understanding of the design and implementation of v-BCI and a-BCI systems, making it easier for BCI researchers to incorporate existing technologies from traditional communication into these systems. Second, it facilitates the comparison between v-BCI and a-BCI as well as between systems using different EEG signals. Third, it can help to transfer the existing methodologies and techniques in communication systems to improve system performance of the current v-BCI and a-BCI systems.

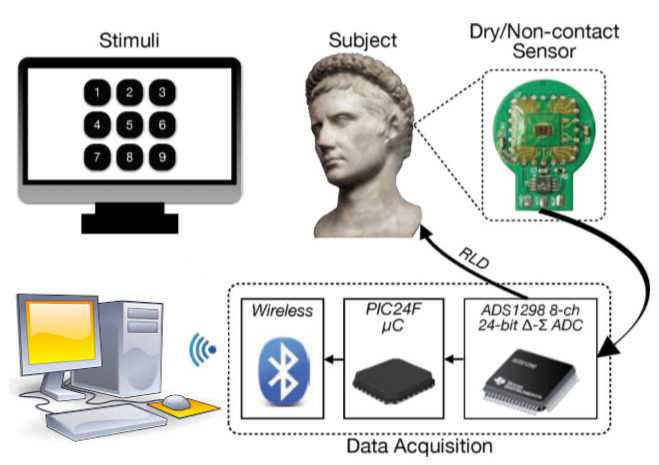
**2.2.2 Dry and Noncontact EEG Sensors for Mobile Brain–Computer Interface**

Dry and noncontact electroencephalographic (EEG) electrodes, which do not require gel or even direct scalp coupling, have been considered as an enabler of practical, real-world, brain–computer interface (BCI) platforms. This study compares wet electrodes to dry and through hair, noncontact electrodes within a steady state visual evoked potential (SSVEP) BCI paradigm. The construction of a dry contact electrode, featuring ﬁgered contact posts and active buffering circuitry is presented. Additionally, the development of a new, noncontact, capacitive electrode that utilizes a custom integrated, high-impedance analog front-end is introduced. Figure 2.1 shows the performance of Wet, Dry and Non contact electrodes.



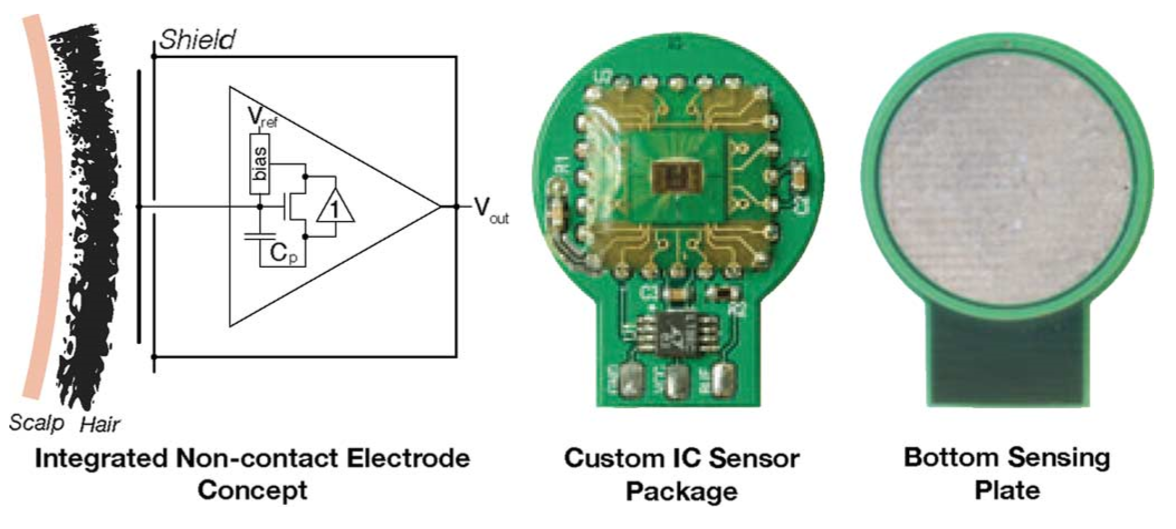
**Figure 2.1 Comparison of performance between Different Electrodes**

The data from the noncontact electrode, operating on the top of hair, show a maximum ITR in excess of 19 bits/min at 100% accuracy (versus 29.2 bits/min for wet electrodes and 34.4 bits/min for dry electrodes), a level that has never been demonstrated before. The results of these experiments show that both dry and noncontact electrodes, with further development, may become a viable tool for both future mobile BCI and general EEG applications. The dry sensor consists of two sections. A lower plate contains a set of spring-loaded pin contacts which can easily penetrate hair without the need for any preparation. The gold plated “ﬁngers” achieve direct electrical connection to the scalp. A male snap connector (identical to the one used for ECG electrodes) on the top side of the plate mates with it’s female counter part on a second PCB which contains the active electrode circuitry. Relatively high impedance signals offered by the dry contact are buffered with an off-the-shelf CMOS-input opamp (National Semiconductor LMP7702). The unity gain buffer, along with the shielded cabling, greatly reduces the effects of external interference.



**Figure 2.2 General Architecture of BCI system**

The new integrated sensor achieves it’s high input impedance through careful design and control of the sensitive input node, made possible by the custom VLSI circuit implementation. Figure 2.2 shows us the picture of the integrated noncontact electrode which operates on top of hair. The integrated electrode achieves input impedances much greater than what has been possible with discrete designs through careful shielding and custom packaging made possible with a fully custom IC design. The figure 2.3 shows the Custom Integrated Circuit Non-Contact Electrode’s design.



**Figure 2.3 Custom IC Non-Contact Electrodes**

Prince Mahajan et al (2009) describes the design, implementation, and evaluation of Depot, a cloud storage system that minimizes trust assumptions. Depot tolerates buggy or malicious behaviour by any number of clients or servers, yet it provides safety and guarantees to correct clients. Depot provides these guarantees using two-layer architecture. First, Depot ensures that the updates observed by correct nodes are consistently ordered under Fork-Join-Causal consistency (FJC). Second, Depot implements protocols that use this consistent ordering of updates to provide other desirable consistency, staleness, durability, and recovery properties.

Stephen Manuel Wolfson (2008) discussed the possible existence of a domestic surveillance/data collection program conducted by the National Security Agency (“NSA”) with the assistance of AT&T, and the implications of such a program under the Electronic Communications Privacy Act (“ECPA”). Next, it turns to the story of former AT&T technician Mark Klein and the Electronic Frontier Foundation’s (“EFF”) case, Hepting v. AT&T Corporation. Klein claims that the NSA has built a “secret room” in AT&T’s San Francisco switching center that grants the agency access to a vast amount of customer information.

**2.3 ISSUES IN THE EXISTING SYSTEMS**

* P300 algorithms are based only on attentiveness of a person which accounts for binary control only.
* Concentration is used as the only parameter and hence time complexity is high.
* Any operation performed using the system is restricted to creation of textual strings rather than execution of any sort.

**2.4 PROPOSED SYSTEM**

The proposed system scheme consists of four phases, namely:

* + Signal Extraction, Filtering and Transmission
  + Raw data Processing
  + Virtual Interface
  + Device Controller

**2.5 SUMMARY**

The literature survey documents are presented in this chapter. These documents describe the existing features of the project and the issues in it. The proposed system is described to overcome the issues in existing system.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 INTRODUCTION**

System design is the process or art of defining the architecture, components, modules, interfaces and date for a system to satisfy specified requirements. The main purpose of the design phase is to plan a solution for the problem specified by the required document. The design phase takes as input, the requirements in Requirement Analysis stage. For each requirement, a set of one or more design elements will be produced as a result. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance. In system design, the design functions and operations are described in detail, including the screen layouts, process diagrams and other documentation. The output of this phase is the functional design document and will describe the new system as a collection of modules or subsystems.

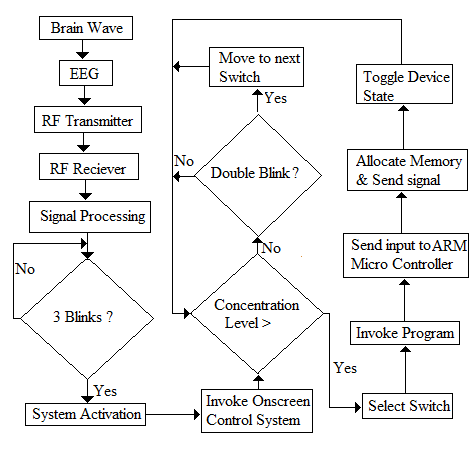
The design documents created for the proposed system consists of a system architecture, functional architecture and algorithms which aims to identify the modules that should be in the system, the specification of the modules, and how they interact with each other to produce the desired results. The system architecture deals with the physical components and their interaction for the proposed system. The functional architecture defines the functionality of each module and their logical interaction. The algorithm explains stepwise procedure of each module.

**3.2 SYSTEM ARCHITECTURE**

**Figure 3.1 System Architecture**

The system architecture of our proposed system is shown in Figure 3.1. The input to the system is the brain wave extracted from the brain using a brain sensor and then transmitted to the processing unit. In the processing unit noises are removed and we analysis the brain waves looking for patterns which corresponds to eye blink and also attention levels. From the output of the processing unit, a graph is plotted based on which we may control the devices via a micro controller. Thus we can control any device using our brainwave.

**3.3 FUNCTIONAL ARCHITECTURE**



**Figure 3.2 Functional Architecture**

A functional architecture is an architectural model that identifies enterprise functions, interactions and corresponding IT needs. These functions can be used as reference by different domain experts to develop IT-systems as part of a co-operative information-driven enterprise. The functional architecture of the proposed system is depicted in the Figure 3.2.

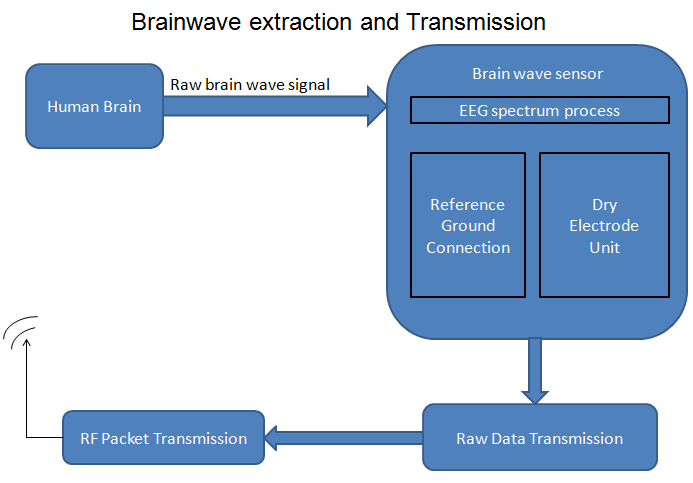
**3.4 MODULAR DESIGN**

Modularity is a general system concept typically defined as a degree to which a system’s components may be separated and recombined. The proposed system consists of following modules.

1. Brainwave Extraction and Transmission
2. Raw Data Processing
3. Visual Interface
4. Device Controller

**3.4.1 Brainwave Extraction and Transmission**

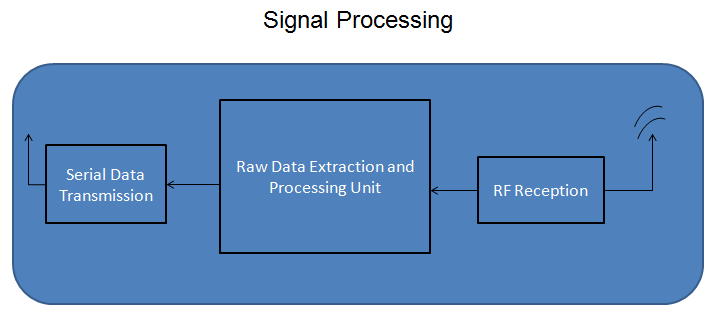
Figure 3.3 depicts the Brainwave extraction and Transmission module. The sensor is placed in the target’s head and the brain waves are recorded using this sensor. The brain wave sensor consists of EEG spectrum process, reference ground connection and a dry electrode unit(A dry electrode reads the brain signals without applying any gel in the scalp of the target). Once the signal is obtained from the brain, the raw data is transmitted with the help of radio frequency transmission.

****

**Figure 3.3 Brainwave extraction and Transmission module**

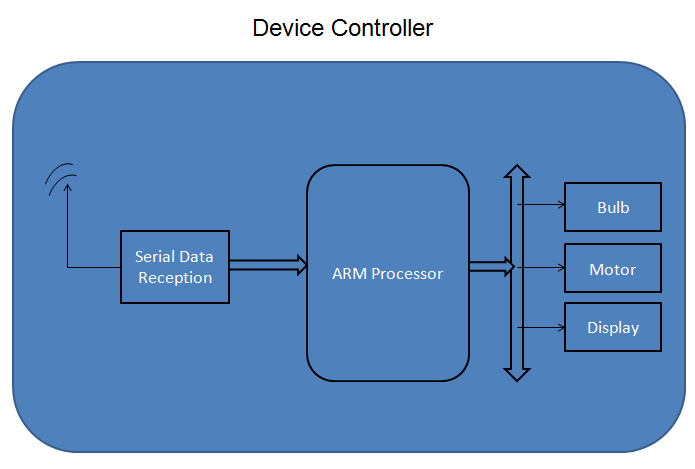
**3.4.2 Raw Data Processing**

The raw data transmitted is received and fed into the matlab for processing. In matlab we install Brain Computer Interface tools so as to support our needs. The continuous stream of raw data will be given as the input to matlab. The brain waves collected are examined for patterns indicating a forced eye blink or a spike in attention level. We fix a threshold which differentiates normal eye blink of the user with that of a forced eye blink. This also filters the unnecessary details like noise, disturbance etc. The main aim of this module is to find the occurrence of double and triple blinks and also to track the attention level of the user. This process is pictographically represented in Figure 3.4



**Figure 3.4 Raw Data Processing**

**3.4.3 Home Appliance Controller**

****

**Figure 3.5 Home Appliance Controller**

Figure 3.5 depicts the device Controller phase. If a triple blink is detected the system is turned on and from here on in, we wait for a double blink or the attention level to go high than normal. A double blink indicates the movement of current focus to next button and a hike in attention level invoke the micro controller code responsible for specified device control. .

**3.5 SYSTEM REQUIREMENTS**

**3.5.1 Hardware Requirements**

RAM : 1 GB and above

Processor : Dual core and above

Hard Disk : 80 GB and above

Brainwave Sensor : Neurosky

Microcontroller : ARM LPC 2148

Other Devices : Bluetooth Extension

**3.5.2 Software Requirements**

Operating system : Windows 7 and above

Language : Java

Embedded C : KEIL Compiler, Flash Magic

Simulation Software : Matlab

**3.6 SUMMARY**

The functional architecture of the individual modules has been designed and detailed description of each of the processes has been given. This gives us an overall view of the system at a lower level and acts as a source of reference for the design phase. The relationship between the module and processes has been clearly depicted.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.1 INTRODUCTION**

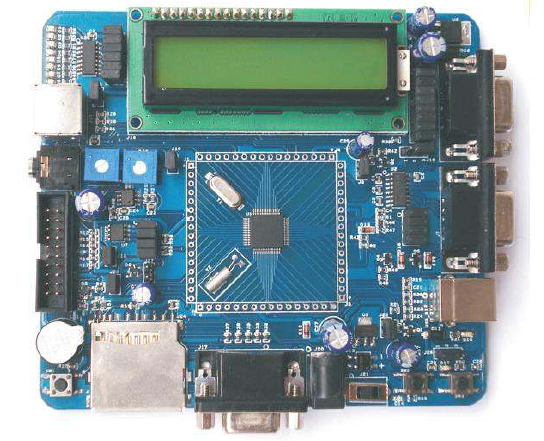
This chapter discusses the actual implementation of the project in depth. The second section gives a brief account of the platform, which is ASP.NET, on which the implementation is done. The third section describes the functions that were used in the implementation. Snapshots of the obtained results have been included to describe the processing. The last section concisely summarizes the whole chapter.

**4.2 OVERVIEW OF THE PLATFORM**

**4.2.1 Microcontroller**

**4.2.1.1 LPC 2148 Board**

The LPC2141/2/4/6/8 microcontrollers are based on a 32/16 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and an unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty. Due to their tiny size and low power consumption, LPC2141/2/4/6/8 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTs, SPI, SSP to I2Cs, and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.



**Figure 4.1 LPC 2148 Board**

**4.2.1.2 Features**

* 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
* 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory. 128 bit wide interface/accelerator enables high speed 60 MHz operation.
* In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.
* Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip RealMonitor software and high speed tracing of instruction execution.
* USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provide 8 kB of on-chip RAM accessible to USB by DMA.
* One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14 analog inputs, with conversion times as low as 2.44 μs per channel.
* Single 10-bit D/A converter provides variable analog output.
* Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
* Low power real-time clock with independent power and dedicated 32 kHz clock input.
* Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus (400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
* Vectored interrupt controller with configurable priorities and vector addresses.
* Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
* Up to nine edge or level sensitive external interrupt pins available.
* 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 μs.
* On-chip integrated oscillator operates with an external crystal in range from 1 MHz to 30 MHz and with an external oscillator up to 50 MHz.
* Power saving modes include Idle and Power-down.
* Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
* Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).
* Single power supply chip with Power-On Reset (POR) and BOD circuits: – CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

**4.2.1.3** **Applications**

* Industrial control
* Medical systems
* Access control
* Point-of-sale
* Communication gateway
* Embedded soft modem
* General purpose applications

**4.2.1.4 ARM7TDMI-S processor**

The ARM7TDMI-S is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler than those of microprogrammed Complex Instruction Set Computers. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor core.

Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory. The ARM7TDMI-S processor also employs a unique architectural strategy known as THUMB, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The key idea behind THUMB is that of a super-reduced instruction set. Essentially, the ARM7TDMI-S processor has two instruction sets:

* The standard 32-bit ARM instruction set.
* A 16-bit THUMB instruction set.

The THUMB set’s 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM’s performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because THUMB code operates on the same 32-bit register set as ARM code. THUMB code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system. The ARM7TDMI-S processor is described in detail in the ARM7TDMI-S Datasheet that can be found on official ARM website.

**4.2.1.5 On-chip flash memory system**

The LPC2141/2/4/6/8 incorporates a 32 kB, 64 kB, 128 kB, 256 kB, and 512 kB Flash memory system, respectively. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways: over the serial built-in JTAG interface, using In System Programming (ISP) and UART0, or by means of In Application Programming (IAP) capabilities. The application program, using the IAP functions, may also erase and/or program the Flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc. When the LPC2141/2/4/6/8 on-chip bootloader is used, 32 kB, 64 kB, 128 kB, 256 kB, add 500 kB of Flash memory is available for user code. The LPC2141/2/4/6/8 Flash memory provides minimum of 100,000 erase/write cycles and 20 years of data-retention.

**4.2.1.6 On-chip Static RAM (SRAM)**

On-chip Static RAM (SRAM) may be used for code and/or data storage. The on-chip SRAM may be accessed as 8-bits, 16-bits, and 32-bits. The LPC2141/2/4/6/8 provide 8/16/32 kB of static RAM, respectively.

The LPC2141/2/4/6/8 SRAM is designed to be accessed as a byte-addressed memory. Word and halfword accesses to the memory ignore the alignment of the address and access the naturally-aligned value that is addressed (so a memory access ignores address bits 0 and 1 for word accesses, and ignores bit 0 for halfword accesses). Therefore valid reads and writes require data accessed as halfwords to originate from addresses with address line 0 being 0 (addresses ending with 0, 2, 4, 6, 8, A, C, and E in hexadecimal notation) and data accessed as words to originate from addresses with address lines 0 and 1 being 0 (addresses ending with 0, 4, 8, and C in hexadecimal notation). This rule applies to both off and on-chip memory usage. The SRAM controller incorporates a write-back buffer in order to prevent CPU stalls during back-to-back writes. The write-back buffer always holds the last data sent by software to the SRAM. This data is only written to the SRAM when another write is requested by software (the data is only written to the SRAM when software does another write). If a chip reset occurs, actual SRAM contents will not reflect the most recent write request. Any software that checks SRAM contents after reset must take this into account. Two identical writes to a location guarantee that the data will be present after a Reset. Alternatively, a dummy write operation before entering idle or power-down mode will similarly guarantee that the last data written will be present in SRAM after a subsequent reset.

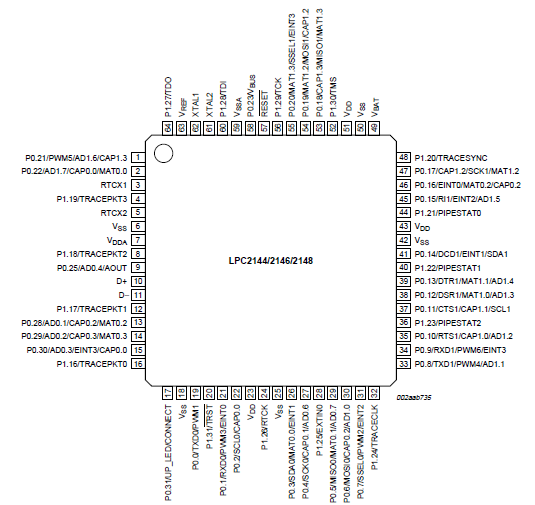
**4.2.1.7 System Control Block**

The System Control Block includes several system features and control registers for a number of functions that are not related to specific peripheral devices. These include:

* Crystal Oscillator
* External Interrupt Inputs
* Miscellaneous System Controls and Status
* Memory Mapping Control
* PLL
* Power Control
* Reset
* APB Divider
* Wakeup Timer

Each type of function has its own register(s) if any are required and unneeded bits are defined as reserved in order to allow future expansion. Unrelated functions never share the same register addresses

**4.2.1.8 Pin Configuration:**



**Figure 4.2 Pin diagram of ARM processor**

The pin connect block allows selected pins of the microcontroller to have more than one function. Configuration registers control the multiplexers to allow connection between the pin and the on chip peripherals. Peripherals should be connected to the appropriate pins prior to being activated, and prior to any related interrupt(s) being enabled. Selection of a single function on a port pin completely excludes all other functions otherwise available on the same pin. The only partial exception from the above rule of exclusion is the case of inputs to the A/D converter. Regardless of the function that is selected for the port pin that also hosts the A/D input, this A/D input can be read at any time and variations of the voltage level on this pin will be reflected in the A/D readings. However, valid analog reading(s) can be obtained if and only if the function of an analog input is selected. Only in this case proper interface circuit is active in between the physical pin and the A/D module.

**4.2.2 Keil Compiler**

**4.2.2.1 Keil IDE’s:**

* This tool is used to develop the source code needed for the design.
* The tool helps us not only to develop but also compile the code and simulate the code.
* The keil tool is also used to convert the compiled Embedded C code to its equivalent hex code.

**4.2.2.2 Keil C Compiler:**

Keil Software publishes one of the most complete development tool suites for 8051 software, which is used throughout industry. For development of C code, their Developer's Kit product includes their C51 compiler, as well as an integrated 8051 simulator for debugging. A demonstration version of this product is available on their website, but it includes several limitations.

The C programming language was designed for computers, though, and not embedded systems. It does not support direct access to registers, nor does it allow for the reading and setting of single bits, two very important requirements for 8051 software. In addition, most software developers are accustomed to writing programs that will by executed by an operating system, which provides system calls the program may use to access the hardware. However, much code for the 8051 is written for direct use on the processor, without an operating system. To support this, the Keil compiler has added several extensions to the C language to replace what might have normally been implemented in a system call, such as the connecting of interrupt handlers.

The purpose of this manual is to further explain the limitations of the Keil compiler, the modifications it has made to the C language, and how to account for these in developing software for the 8051 microcontroller.

**4.2.2.3 Keil Limitations**

* There are several very important limitations in the evaluation version of Keil's Developer's Kit that users need be aware of when writing software for the 8051.
* Object code must be less than 2 Kbytes.
* The compiler will compile any-sized source code file, but the final object code may not exceed 2 Kbytes. If it does, the linker will refuse to create a final binary executable (or HEX file) from it. Along the same lines, the debugger will refuse any files that are over 2Kbytes, even if they were compiled using a different software package.
* Few student projects will cross this 2Kbyte threshold, but programmers should be aware of it to understand why code may no longer compile when the project grows too large.
* Program code starts at address 0x4000.
* All C code compiled and linked using the Keil tools will begin at address 0x4000 in code memory. Such code may not be programmed into devices with less than 16Kbytes of Read-Only Memory. Code written in assembly may circumvent this limitation by using the "origin" keyword to set the start to address 0x0000. No such work-around exists for C programs, though. However, the integrated debugger in the evaluation software may still be used for testing code. Once tested, the code may be compiled by the full version of the Keil software, or by another compiler that supports the C extensions used by Keil.

**4.2.2.4 C Modifications**

The Keil C compiler has made some modifications to another wise ANSI-compliant implementation of the C programming language. These modifications were made solely to facilitate the use of a higher-level language like C for writing programs on microcontrollers.

**Variable Types**

The Keil C compiler supports most C variable types and adds several of its own.

**Standard Types**

The evaluation version of the Keil C compiler supports the standard ANSI C variable types, with the exception of the floating-point types. These types are summarized below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Types** | **Bits** | **Bytes** | **Range** |
| Char | 8 | 1 | -128 to +127 |
| Unsigned char | 8 | 1 | 0 to 255 |
| Enum | 16 | 2 | -32,768 to +32,767 |
| Short | 16 | 2 | -32,768 to +32,767 |
| Unsigned short | 16 | 2 | 0 to 65,535 |
| Int | 16 | 2 | -32,768 to +32,767 |
| Unsigned int | 16 | 2 | 0 to 65,535 |
| Long | 32 | 4 | -2,147,483,648 to +2,147,483,647 |
| Unsigned long | 32 | 4 | 0 to 4,294,697,295 |

**Table 4.1 Standard Types**

In addition to these variable types, the compiler also supports the struct and union data structures, as well as type redefinition using typedef.

**Keil Types**

To support a microcontroller and embedded systems applications, Keil added several new types to their compiler. These are summarized in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Bits** | **Bytes** | **Range** |
| Bit | 1 | 0 | 0 to 1 |
| Sbit | 1 | 0 | 0 to 1 |
| Sfr | 8 | 1 | 0 to 255 |
| sf16 | 16 | 2 | 0 to 65,535 |

**Table 4.2 Keil Types**

Of these, only the bit type works as a standard variable would. The other three have special behavior that a programmer must know.

‘Bit’ is a data type that gets allocated out of the 8051's bit-addressable on-chip RAM. Like other data types, it may be declared as either a variable. However, unlike standard C types, if may not be used as a pointer.

‘Sbit, sfr, and sf16’ are special types for accessing 1-bit, 8-bit, and 16-bit special function registers. Because there is no way to indirectly address registers in the 8051, addresses for these variables must be declared outside of functions within the code. Only the data addressed by the variable may be manipulated in the code.

Conveniently, the standard special function registers are all defined in the reg51.h file that any developer may include into their source file. Only registers unique to the particular 8051-derivative being used for the project need have these variable declared, such as registers and bits related to a second on-chip serial port.

**Keil Variable Extensions**

In writing applications for a typical computer, the operating system handles manages memory on behalf of the programs, eliminating their need to know about the memory structure of the hardware. Even more important, most computers having a unified memory space, with the code and data sharing the same RAM. This is not true with the 8051, which has separate memory spaces for code, on-chip data, and external data.

To accommodate for this when writing C code, Keil added extensions to variable declarations to specify which memory space the variable is allocated from, or points to. The most important of these for student programmers are summarized in the following table.

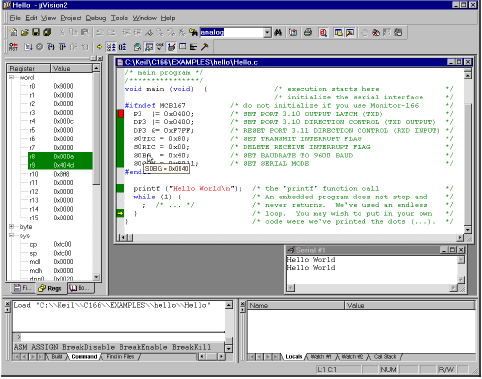
|  |  |  |
| --- | --- | --- |
| **Extension** | **Memory Type** | **Related ASM** |
| Data | Directly-addressable data memory (data memory addresses 0x00-0x7F) | MOV A, 07Fh |
| Idata | Indirectly-addressable data memory (data memory addresses 0x00-0xFF) | MOV R0, #080h MOV A, R0 |
| Xdata | External data memory | MOVX @DPTR |
| Code | Program memory | MOVC @A+DPTR |

**Table 4.3 Keil Variable Extensions**

These extensions may be used as part of the variable type in declaration or casting by placing the extension after the type, as in the example below. If the memory type extension is not specified, the compiler will decide which memory type to use automatically, based on the memory model.

**Keil Function Extensions**

Keil provides two important extensions to the standard function declaration to allow for creation of interrupt handlers and reentrant functions.

****

**Figure 4.3 Sample Program**

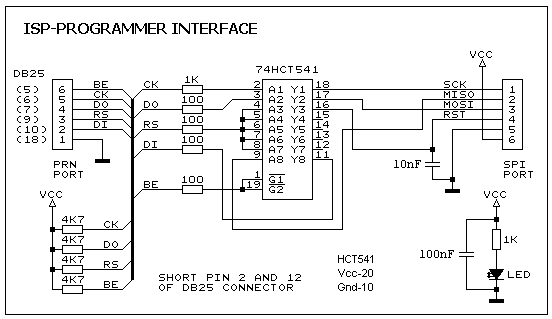
**4.2.3 Flash Programmer**

Flash programmer is used to fuse the built hex code into the Microcontroller Language using ‘Embedded C’ coding.

**4.2.3.1 Introduction**

This ISP Programmer can be used either for in-system programming or as a stand-alone SPI programmer for Atmel ISP programmable devices. The programming interface is compatible to STK200 ISP programmer hardware so the users of STK200 can also use the software which can program both the 8051 and AVR series devices.

**4.2.3.2 Hardware**

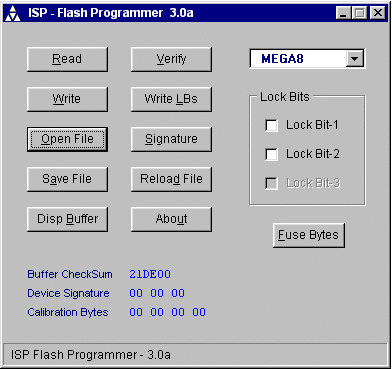
****

**Figure 4.4 Circuit of the in-system programmer interface**

The power to the interface is provided by the target system. The 74HCT541 IC isolate and buffer the parallel port signals. It is necessary to use the HCT type IC in order to make sure the programmer should also work with 3V type parallel port.

**4.2.3.3 Software**

The ISP-30a.zip file contains the main program and the I/O port driver. Place all files in the same folder. The main screen view of the program is shown in figure 4.5. Also make sure do not program the RSTDISBL fuse in ATmega8, ATtiny26 and ATtiny2313 otherwise further SPI programming is disable and you will need a parallel programmer to enable the SPI programming. For the fuses setting consult the datasheet of the respective device. For the auto hardware detection, it is necessary to short pin 2 and 12 of DB25 connector, otherwise the software uses the default parallel port i.e. LPT1.



**Figure 4.5 Main screen of the program ISP-Program Version 3.0**

**4.2.4 MATLAB**

MATLAB (Matrix Laboratory) is a multi-paradigm [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](http://en.wikipedia.org/wiki/Fourth-generation_programming_language). Developed by [Math Works](http://en.wikipedia.org/wiki/MathWorks), MATLAB allows [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) manipulations, plotting of [functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of [algorithms](http://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](http://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including C, C++, [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and [Fortran](http://en.wikipedia.org/wiki/Fortran). MATLAB is widely used in academic and research institutions as well as industrial enterprises. MATLAB was first adopted by researchers and practitioners in [control engineering](http://en.wikipedia.org/wiki/Control_engineering). It is now also used in education, in particular the teaching of [linear algebra](http://en.wikipedia.org/wiki/Linear_algebra) and [numerical analysis](http://en.wikipedia.org/wiki/Numerical_analysis), and is popular amongst scientists involved in [image processing](http://en.wikipedia.org/wiki/Image_processing). The MATLAB application is built around the MATLAB language. MATLAB is used in vast area, including signal and image processing, communications, control design, [test and measurement](http://www.mathworks.in/applications/t_m), financial modelling and analysis, and computational.

The following are the features of MATLAB:

* Interactive tools for iterative exploration, design, and problem solving.
* Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
* 2-D and 3-D graphics functions for visualizing data.
* Tools for building custom graphical user interfaces.
* High level language for technical computing
* Development environment for managing code, files and data

MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, [object-oriented programming](http://www.mathworks.in/products/matlab/object_oriented_programming.html) (OOP), and debugging features. MATLAB lets you execute commands or groups of commands one at a time, without compiling and linking, enabling you to quickly iterate to the optimal solution. For fast execution of heavy matrix and vector computations, MATLAB uses processor-optimized libraries. For general-purpose scalar computations, MATLAB generates machine-code instructions using its JIT (Just-In-Time) compilation technology. MATLAB supports the entire data analysis process, from acquiring data from external devices and databases, through pre-processing, visualization, and numerical analysis, to producing presentation-quality output.

**4.2.5 Visual Basic**

* Visual Basic is a programming language and development environment created by Microsoft.
* Visual Basic provides a graphical user interface GUI that allows the developer drag and drop objects into the program as well as manually write program code.
* Visual Basic, also referred to as "VB," is designed to make software development easy and efficient

**4.2.5.1 Visual Basic 6.0**

* Visual Basic is a programming language and integrated development environment.
* It derives from the much older BASIC programming language, and so is considered useful and easy programming language for the beginner to learn.
* Visual Basic 6.0 was the final edition of Visual Basic.

**4.2.5.2 Features**

Learning Consists of all necessary tools required to build main stream windows applications.

* Professional Includes advanced features such as tools to develop ActiveX and Internet controls.
* Enterprise In addition to all Professional features, it also includes tools such as Visual
* **GUI Interface:** VB is a Graphical User Interface language. This means that a VB program will always show something on the screen that the user can interact with to get a job done.
* **Modularization:** It is considered good programming practice to modularize your programs. Small modules where it is clearly indicated what comes into the module and what goes out makes a program easy to understand.
* **Object Oriented:** Object Oriented Programming is a concept where the programmer thinks of the program in "objects" that interact with each other. Visual Basic forces this good programming practice.
* **Debugging:** Visual Basic offers two different options for code debugging.
  + The Debugging Managed Code individually debugs C and C++ applications and Visual Basic Windows applications.
  + The Runtime Debugger helps to find and fix bugs in programs at runtime.

* **Data Access Feature:** By using data access features, we can create databases, scalable server-side components for most databases, including Microsoft SQL Server and other enterprise-level database.
* **Macros IDE:** The Macros integrated development environment is similar in design and function to the Visual Studio IDE. The Macros IDE includes a code editor, tool windows, the properties windows and editors.

**4.3 IMPLEMENTATION DETAILS**

**4.3.1 Brainwave Extraction and Transmission**

****

**4.6 Neurosky EEG Headset**

Figure 4.6 represents the EEG headset from Neurosky. In the project we are using this particular device to extract brainwaves from the brain. The headset is based on a dry electrode sensor (as shown in Figure 4.7) which is placed at strategic portion of the head in order to extract the desired waves.

****

**4.7 EEG Sensor of the headset**

By extracting the desired waves we are able to determine the response of the person using the BCI system. The extracted wave is sent as data stream to a PC via an RF transmitter (Bluetooth transimission).

**4.3.2 Raw Data Processing**

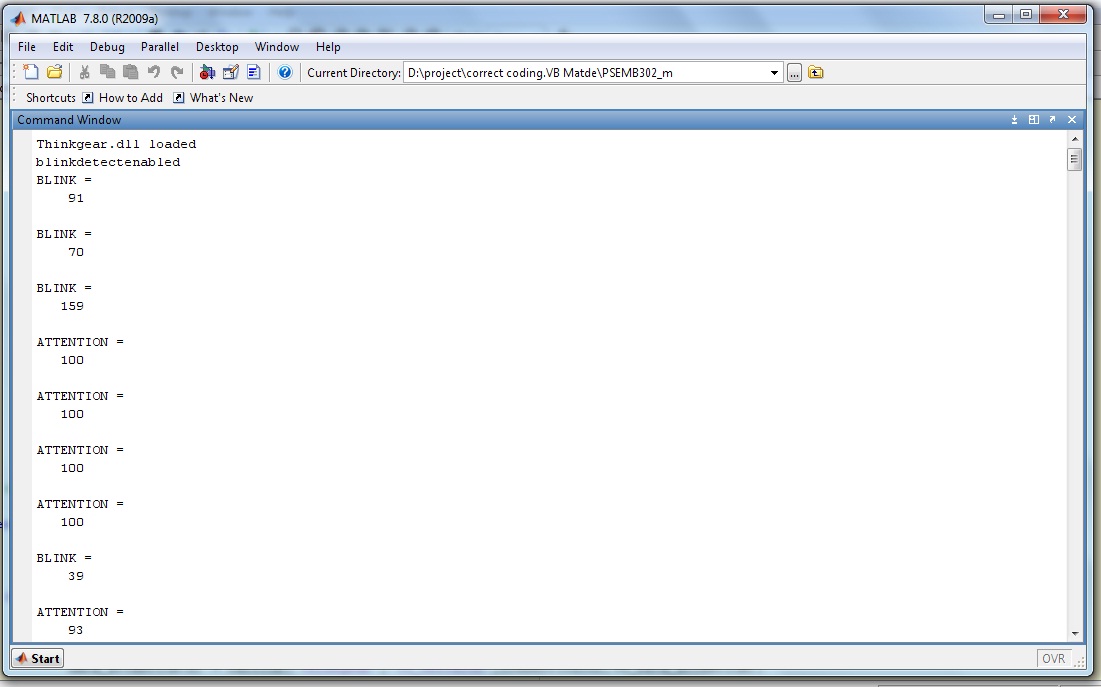
The data stream received in the PC is reconstructed into a wave which can be used for further analysis. Figure 4.8 shows us the Brainwave Visualizer which helps us visually see the reconstructed brain wave and also allow us to recognize different parts of the wave corresponding to different activities performed by the user.



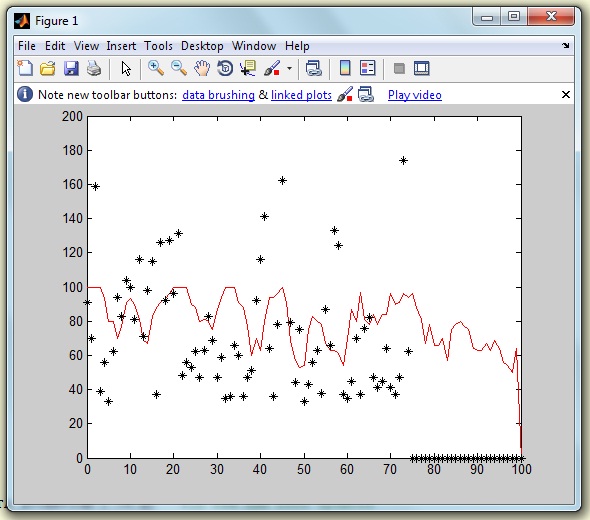
**4.8 Brainwave Visualizer**

Though the brainwave Visualizer has the ability to reconstruct waves and differentiate the wave, it is not used in any part of the implementation. The brain wave Visualizer is useful in finding out the optimal “Attention and Meditation levels” of a user and hence these readings are useful in calibrating the system for optimal performance for that particular user.

These data streams are received by Matlab which is where the real execution takes place. The Matlab program constantly records the Attention level and the blink movement for the subject and records them as integer values. In Figure 4.9, we can observe that in Matlab, various frequencies of the waves are analyzed and hence it is finally sorted into different levels of attention and blink respectively. Based on these levels, a graph is plotted (as shown in Figure 4.10) which gives us a detailed view of past and present concentration level and eye blink of the subject from since the system is online.

****

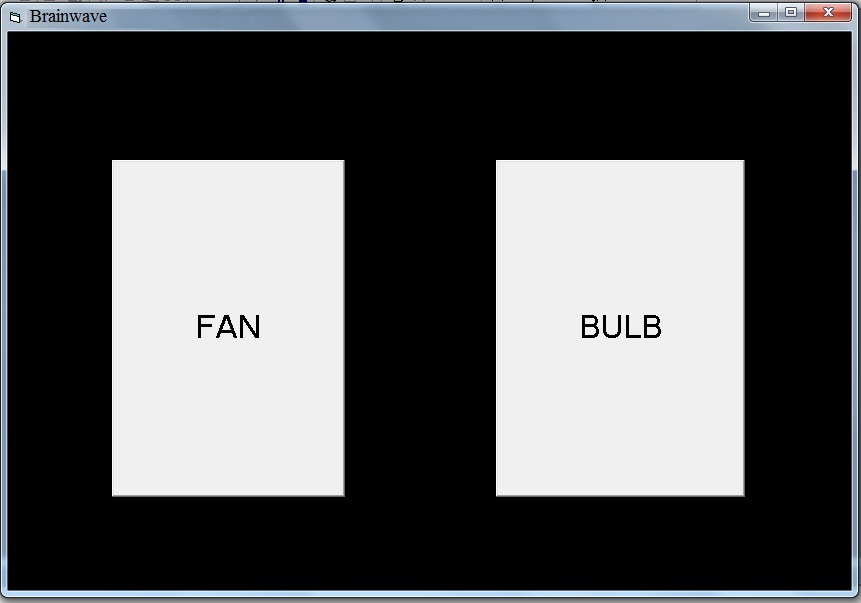
**Figure 4.9 Matlab readings**

****

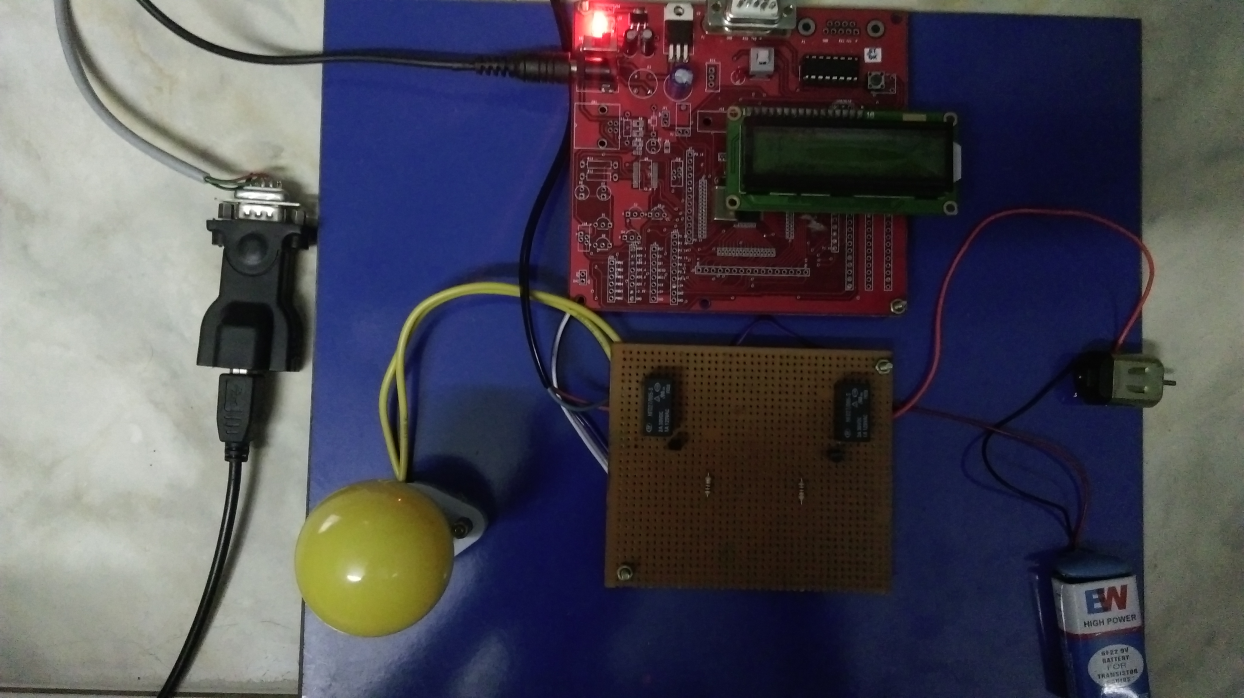
**Figure 4.10 Graph plotted from Subject**

**4.3.3 Device Controller**

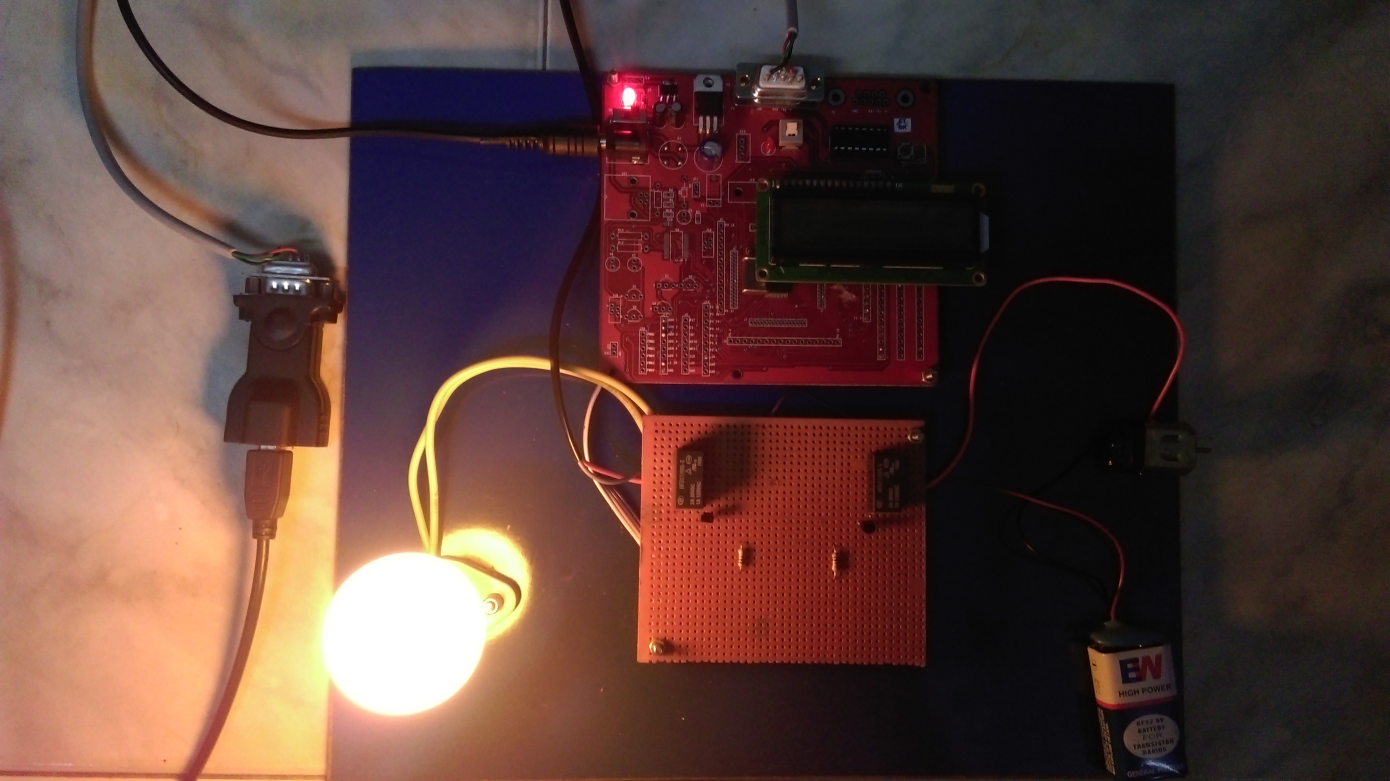
After the waves have been processed by Matlab, the readings are used to control the Visual System (As seen in Figure 4.11). By using blink and attention levels, a user may control the visual system. As the user selects an option, the program invokes the corresponding Embedded C program which is embedded in a Microprocessor connected through a USB to Serial Convertor (as shown in Figure 4.12). Depending on the Choice of selection the microprocessor trips the relay thus switching on or off the device required.



**Figure 4.11 Virtual Switch**



**Figure 4.12 Electronic Setup**



**Figure 4.13 Electronic Setup in ‘ON’ State**

**4.4 SUMMARY**

This chapter has brought out the implementation details of the proposed system. It describes the working of all the process of our proposed system using Matlab and visual basic 6. This chapter also described the overview of platform. In this chapter snapshots of different scenarios at different instants are also presented.

**CHAPTER 5**

**RESULT AND PERFORMANCE ANALYSIS**

**5.1 INTRODUCTION**

Testing is a schedule process carried out by the software development team to capture all the possible errors, missing operations and also a complete verification to verify objectives are met and the user requirement are satisfied. The design of test for software and other engineering products can be as challenging as the initial design to the product itself.

**5.2 TESTING TYPES**

A software engineering product can be tested in one of the following ways:

* Black box testing
* White box testing

**5.2.1 Black Box Testing**

Knowing the specified function that product has been designed to perform, determine whether each function is fully operational.

**5.2.2 White Box Texting**

Knowing the internal workings of a software product determine whether the internal operation implementing the function perform ac have been according to the specification and all the internal components have been adequately exercised.

**5.3 TESTING STRATEGIES**

Testing strategies that are often adopted by the software by the software development team include:

**5.3.1 Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic functioning properly, and that program input produces valid outputs. Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. This is a structural testing, that relies on knowledge of its construction and it’s invasive.

**5.3.2 Integration Testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as testing is specially aimed at exposing the problems that arise from the combination of components.

**5.3.3 Functional Testing**

Functional rests provide a systematic demonstration that functions to be tested are available as specified by the business and technical requirements, system documentation, and user manuals etc.

**5.3.4 System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven links and integration points.

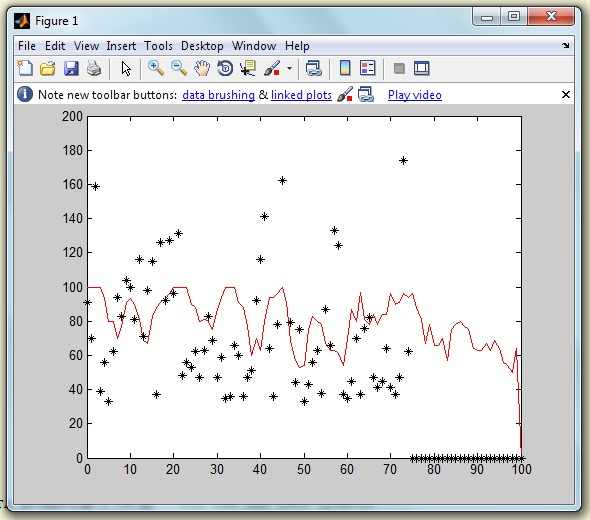
**5.4 TESTING SCHEDULES**

Below are the results of various test cases and their corresponding graphs. In the graph x axis represents time and y axis represents attention and blink values.

**5.4.1 TEST CASE 1**

|  |  |
| --- | --- |
| Test case description | Subject devoid of social contact |
| Type of testing | Unit testing |
| Expected Result | Normal attention level observed throughout the experiment with high peaks at expected instants |
| Status | Pass |

**Table 5.1 Test case 1**

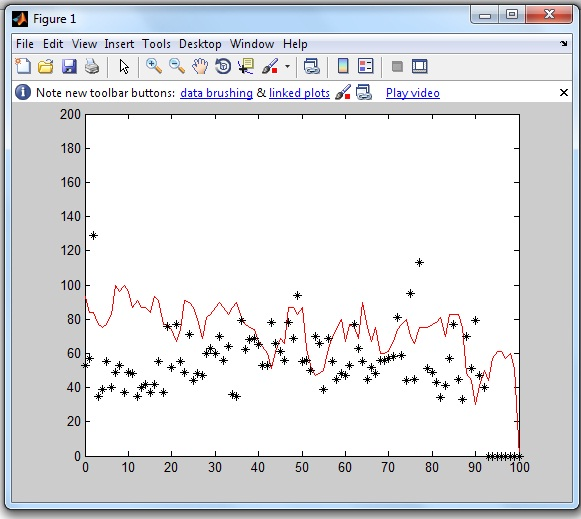


**Figure 5.1 Graph for Test Case 1**

**5.3.2 TEST CASE 2**

|  |  |
| --- | --- |
| Test case description | Subject with several distractions |
| Type of testing | Unit testing |
| Expected Result | High |
| Status | Pass |

**Table 5.2 Test Case 2**

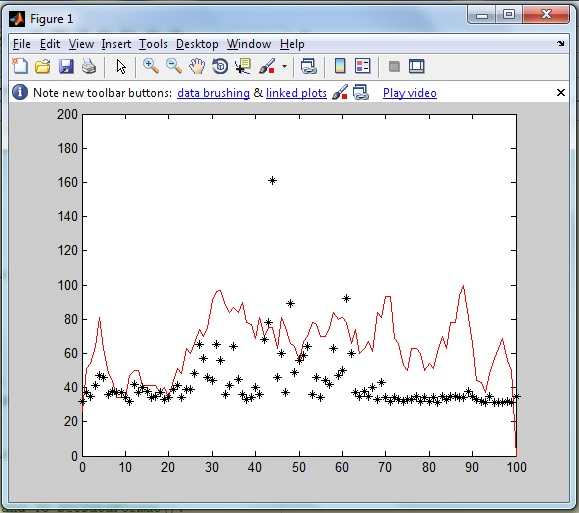


**Figure 5.2 Graph for Test case 2**

**5.3.3 TEST CASE 3**

|  |  |
| --- | --- |
| Test case description | Subject with several distractions |
| Type of testing | Unit testing |
| Expected Result | High |
| Status | Pass |

**Table 5.3 Test Case 3**



**Figure 5.3 Graph for Test Case 3**

**5.5 PERFORMANCE ANALYSIS**

The performance was tested using 30 subjects keeping the threshold for attention and eye blink as 70 and 45 respectively and the efficiency was analyzed at each step of the process. The success rate of the project was found to be approximately 80%. One way to increase the success rate was to alter the threshold values of attention and eye blink. It was found that most of the subjects tend to have normal blink value less than 60 and hence the threshold value for eye blink was set as 60 and above. Similarly by observing the attention values, the threshold was found to be 80. By keeping the above values as threshold for attention and eye blink values the success rate was calculated to be 90 percentages.

**5.6 SUMMARY**

This chapter brought out the test results of various test scenarios. No defects have been registered. All the test cases are executed successfully. This chapter has also described the various testing types and techniques.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 CONCLUSION**

In this paper, we have solved the problem of relating to time complexity and BCI system restricting their execution to textual string creation. Here we had used Embedded C in order to control Microprocessor which helps us achieve control over electronic devices. Further, the above system provides a disabled person autonomous control over his/her surrounding environment thus eliminating the need for a full time nursery care.

**6.2 FUTURE ENHANCEMENTS**

In the future, online learning could be used, in which the classifier is updated after every recorded EEG sample. Accuracy, speed, usability and feedback methods should be improved in the current BCI systems. Accuracy is the most important and affects greatly on the performance of the BCI. Feedback methods could be improved, maybe using games like in the EEG biofeedback. An exhaustive research about the mental tasks could be done in which research topics would include the localization of the brain activity during the mental tasks and how the EEG changes in process of time. If researches in this field continue to flourish then these applications could be improved and BCIs can be used to control a hand or leg prosthesis or a car or even to identify and send the needs of a disabled person to a server which can be processed by an automated robot. For example the need of a disabled person is observed from their brain waves and sent to a robot which takes necessary steps to satisfy their needs.

How well that can be achieved with EEG-based BCIs is not yet known but future seems to be promising. Maybe one day, Non-invasive BCIs recording activity directly from the motor cortex may be used for this kind of purpose in the future which would make BCI systems look seem less.

**APPENDIX**

**SAMPLE SOURCE CODE**

**Brainwave.exe (Form)**

Private Sub BULB\_Click()

MSComm1.Output = "A"

End Sub

Private Sub FAN\_Click()

MSComm1.Output = "B"

End Sub

'Dim dte As Integer

Private Sub Form1\_Load()

With MSComm1

           .CommPort = 1

           .PortOpen = True

           .InputMode = comInputModeText

           .RThreshold = 1

           .Settings = "9600,n,8,1"

           .InputLen = 1

End With

End Sub

**Matlab Code**

portnum1 = 24;   %COM Port of Bluetooth#

comPortName1 = sprintf('\\\\.\\COM%d', portnum1);

// load thinkgear dll

loadlibrary('Thinkgear.dll');

fprintf('Thinkgear.dll loaded\n');

// Get a connection ID handle to ThinkGear

connectionId1 = calllib('Thinkgear', 'TG\_GetNewConnectionId');

if(calllib('Thinkgear','TG\_EnableBlinkDetection',connectionId1,1)==0)

    disp('blinkdetectenabled');

end

import java.util.\*;

import java.awt.event.\*;

mouse.mouseMove(0,0);

screenSize = get(0, 'screensize');

j = 0;

i = 0;

k = 0;

l = 0;

Blink=0;

On\_mode = 0;

count = 0;

mouse\_x=300;

mouse\_y=300;

X = 0:1:255;

while (i < 100)   %loop for 20 seconds

if(data\_BLINK(j) > 45 )

            Blink = Blink+1;

end

if(Blink == 3)

                    Blink=0;

                    On\_mode =1;

                    open('BrainWave.exe');

                   mouse.mouseMove(mouse\_x, mouse\_y);

      end

if(On\_mode == 1)

            if(Blink == 2)

Blink=0;

                        count = count+1;

                        if(count==1)

                            mouse.mouseMove(mouse\_x +400, mouse\_y);

                        end

                        if(count==2)

                            count = 0;

                            mouse.mouseMove(mouse\_x , mouse\_y);

                        end

        end

        end

  if(data\_ATTENTION(k)>70)

  mouse.mousePress(InputEvent.BUTTON1\_MASK)

          mouse.mouseRelease(InputEvent.BUTTON1\_MASK);

end

    plot(X,data\_ATTENTION,'-r',X,data\_BLINK,'\*K');

          axis([0 100 0 200])

    drawnow;

end

// Disconnect

calllib('Thinkgear', 'TG\_FreeConnection', connectionId1 );

end

**REFERENCES**

1. Aleksandra Krolak, Paweł Strumiłło, Springler, (2011), “Eye-blink detection system for human–computer interaction”.
2. M. H. Ang, E. Burdet, C. Guan, B. Rebsamen, C. Teo, C. Wang and H. Zhang, (2010), “A brain controlled wheelchair to navigate in familiar environments”, IEEE Transactions, Neural System, Rehabilitation Engineering, Vol. 18, No. 6.
3. Anthony R. Murphy, Emanuel Donchin, Siri-Maria Kamp (2013), “The Component Structure of Event-Related Potentials in the P300 Speller Paradigm”, IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol.21, No.6.
4. J. Auton, G. Garcia-Molina, A. Nijholt, and T. Tsoneva, (2013), “Emotional brain– computer interfaces”, Int. Adapt. Communication System, Vol. 6, No. 1.
5. Bo Hong, Shangkai Gao, Xiaorong Gao, Yijun Wang (2014) “Visual and Auditory Brain–Computer Interfaces”, , IEEE Transactions On Biomedical Engineering, Vol. 61, No. 5.
6. Bin Xia, Conghui Chen, Dehua An, Hong Xie, Jie Li, (2013 ), “A mental switch-based asynchronous brain computer interface for 2D cursor control”, IEEE Conference.
7. G. E. Birch and S. G. Mason, (2000), “A Brain-Controlled Switch for Asynchronous Control Applications”, IEEE Transactions on Biomedical Engineering, Vol. 47, No. 10.
8. Behrooz Ashtiani, I. Scott MacKenzie, Springler, (2010), “BlinkWrite: efﬁcient text entry using eye blinks”.
9. K. C. C. Chan, H. Leung and L. Zhang, (2008), “Information fusion based smart home control system and its application”, IEEE Transactions Consumer Electron., Vol. 54, No. 3.
10. N. Birbaumer, D. J. McFarland, G. Pfurtscheller, J. R. Wolpaw and T. M. Vaughan, (2000), “Brain-computer interfaces for communication and control”, Clinical Neurophysiology, Vol. 113, No. 6.
11. Charles J. Robinson, Dennis J. McFarland, Emanuel Donchin, Gerwin Schalk. Quatrano, P. Hunter Peckham, Louis A, Jonathan R. Wolpaw, Niels Birbaumer, Theresa M. Vaughan and William J. Heetderks, (2000), “Brain–Computer Interface Technology: A Review of the First International Meeting”, IEEE Transactions on Rehabilitation Engineering, Vol. 8, No. 2.
12. Christoph Maier, Gert Cauwenberghs, Tzyy-Ping Jung, Yijun Wang, Yu Mike Chi, Yu-Te Wang, (2012), “Dry and Noncontact EEG Sensors for Mobile Brain–Computer Interfaces”, IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 20, No. 2.
13. Cong Wang, Jinyi Long, Tianyou Yu, Yuanqing Li, (2013), “A Brain-computer Interface controlled Mail Client”, IEEE Conference.
14. Z. Gu, Y. Li, J. Long and T. Yu, (2012), “Surﬁng the internet with a bci mouse”, Neural Engineering, Vol. 9, No. 3.
15. T.-P. Jung and Y. Wang (2011), “A collaborative brain–computer interface for improving human performance”, Vol.6, No.5.
16. C.Kothe and T.O.Zander, (2011), “Towards passive brain–computer interfaces: Applying brain–computer interface technology to human–machine systems in general”, Neural Engineering, Vol. 8, No. 2.