Building a virtual storage device to life for the secrets

V Ajith[1], P Aravind[2], G David Abishai[3], J Krishna Kumar[4],

Ms.P.Gowthami [5]

[1,2,3,4] Final MEE, [5] Assistant Prof

Medical electronics engineering, Sengunthar college of engineering

Abstract- **This abstract states that the man is called intelligent because of the brain. But we loss the knowledge of a brain when the body is destroyed after the death. Virtual brain project will search for insights into how human beings think and remember. The main aim is to upload human brain into a VBOT Sensor. After the death of the body, the virtual brain will act as the man's brain. Such models will shed light on how memories are stored and retrieved. This could reveal many exciting aspects of the brain, such as the form of memories, memory capacity and how memories are lost. This project contains two sensors wireless body area sensor network named Nanobots.This sensor is a wireless network of wearable computing devices. BAN devices may be embedded inside the body.it gather energy from the body temperature and communicate with the VBOT Sensor. Next Sensor is VBOT Sensor it acts like virtual Brain. Through this sensor we can store our secret and our intelligence with the help of PC or Mobile. We can use the secret of a person after the death.**

Index Terms—Virtual storage device to life for the secrets

I. INTRODUCTION

The man is called intelligent because of the brain. The brain translates the information delivered by the impulses, which then enables the person to react. But we loss the knowledge of a brain when the body is destroyed after the death of man. That knowledge might have been used for the development of the human society. What happen if we create a brain and up load the contents of natural brain into it?

The name of the world’s first virtual brain. That means a machine that can function as human brain. Today scientists are in research to create an artificial brain that can think, response, take decision, and keep anything in memory. The main aim is to upload human brain into machine. So that man can think, take decision without any effort. After the death of the body, the virtual brain will act as the man .So, even after the death of a person we will not loose the knowledge, intelligence, personalities, feelings and memories of that man that can be used for the development of the human society. No one has ever understood the complexity of human brain. It is complex than any circuitry in the world. So, question may arise “Is it really possible to create a human brain?” The answer is “Yes”. Because what ever man has created today always he has followed the nature. When man does not have a device called computer, it was a big question for all. Technology is growing faster than every thing. IBM is now in research to create a virtual brain, called “Blue brain”. If possible, this would be the first virtual brain of the world. With in 30 years, we will be able to scan ourselves into the computers. Is this the beginning of eternal life?

Virtual storage device is an artificial brain, which does not actually the natural brain, but an act as the brain. It can think like brain, take decisions based on the past experience, and response as the natural brain can. It is possible by using a super computer, with a huge amount of storage capacity, processing power and an interface between the human brain and this artificial one. Through this interface the data stored in the natural brain can be up loaded into the computer. So the brain and the knowledge, intelligence of anyone can be kept and used for ever, even after the death of the person. First, it is helpful to describe the basic manners in which a person may be uploaded into a computer. Raymond Kurzweil recently provided an interesting paper on this topic. In it, he describes both invasive and noninvasive techniques. The most promising is the use of very small robots, or nanobots. These robots will be small enough to travel throughout our circulatory systems. Traveling into the spine and brain, they will be able to monitor the activity and structure of our central nervous system. They will be able to provide an interface with computers that is as close as our mind can be while we still reside in our biological form. Nanobots could also carefully scan the structure of our brain, providing a complete readout of the connections between each neuron. They would also record the current state of the brain. This information, when entered into a computer, could then continue to function as us. All that is required is a computer with large enough storage space and processing power. Is the pattern and state of neuron connections in our brain truly all that makes up our conscious selves? Many people believe firmly those we posses a soul, while some very technical people believe that quantum forces contribute to our awareness. But we have to now think technically.

Note, however that we need not know how the brain actually functions, to transfer it to a computer. We need only know the media and contents. The actual mystery of how we achieved consciousness in the first place, or how we maintain it, is a separate discussion. Really this concept appears to be very difficult and complex to us. For this we have to first know how the human brain actually works.

A wireless sensor network (WSN) sometimes called a wireless sensor and actuator network (WSAN) are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

II. SYSTEM OVERVIEW

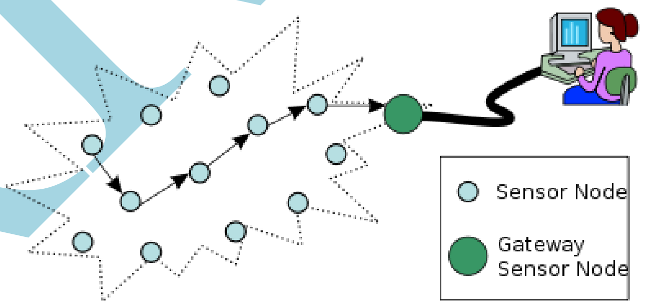
Specification analysis

Today scientists are in research to create an artificial brain that can think, response, take decision, and keep anything in memory. The main aim is to upload human brain into machine. So that man can think, take decision without any effort. After the death of the body, the virtual brain will act as the man .So, even after the death of a person we will not loose the knowledge, intelligence, personalities, feelings and memories of that man that can be used for the development of the human society. No one has ever understood the complexity of human brain. It is complex than any circuitry in the world. So, question may arise “Is it really possible to create a human brain?” The answer is “Yes”. Because what ever man has created today always he has followed the nature. When man does not have a device called computer, it was a big question for all. Technology is growing faster than every thing. IBM is now in research to create a virtual brain, called “Blue brain”. If possible, this would be the first virtual brain of the world. With in 30 years, we will be able to scan ourselves into the computers.

B.Wireless sensor network

A wireless sensor network (WSN) sometimes called a wireless sensor and actuator network (WSAN) are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on. The WSN is built of "nodes" – from a few to several hundreds or even thousands, where each node is connected to one (or sometimes several) sensors. Each such sensor network node has typically several parts: a radio transceiver with an internal antenna or connection to an external antenna, a microcontroller, an electronic circuit for interfacing with the sensors and an energy source, usually a battery or an embedded form of energy harvesting. A sensor node might vary in size from that of a shoebox down to the size of a grain of dust, although functioning "motes" of genuine microscopic dimensions have yet to be created. The cost of sensor nodes is similarly variable, ranging from a few to hundreds of dollars, depending on the complexity of the individual sensor nodes. Size and cost constraints on sensor nodes result in corresponding constraints on resources such as energy, memory, computational speed and communications bandwidth. The topology of the WSNs can vary from a simple star network to an advanced multi-hop wireless mesh network. The propagation technique between the hops of the network can be routing or flooding.

Today we are developed because of our intelligence. Intelligence is the inborn quality that can not be created. Some people have this quality, so that they can think up to such an extent where other can not reach. Human society is always need of such intelligence and such an intelligent brain to have with. But the intelligence is lost along with the body after the death. The virtual brain is a solution to it. The brain and intelligence will alive even after the death. We often face difficulties in remembering things such as people’s names, their birthdays, and the spellings of words, proper grammar, important dates, history, facts etc... In the busy life every one want to be relaxed. Can’t we use any machine to assist for all these? Virtual brain may be the solution to it. What if we upload ourselves into computer, we were simply aware of a computer, or maybe, what if we lived in a computer as a program?



III. HARDWARE

The AVR is a modified Harvard architecture 8-bit RISC single-chip microcontroller, which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM,EPROM, or EEPROM used by other microcontrollers at the time.

The AVR is a modified Harvard architecture machine, where program and data are stored in separate physical memory systems that appear in different address spaces, but having the ability to read data items from program memory using special instructions.

32-bit AVRs

In 2006 Atmel released microcontrollers based on the 32-bit AVR32 architecture. They include SIMD and DSP instructions, along with other audio- and video-processing features. This 32-bit family of devices is intended to compete with the ARM-based processors. The instruction set is similar to other RISC cores, but it is not compatible with the original AVR or any of the various ARM cores.

Device architecture

Flash, EEPROM, and SRAM are all integrated onto a single chip, removing the need for external memory in most applications. Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices. Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

Program memory

Program instructions are stored in non-volatile flash memory. Although the MCUs are 8-bit, each instruction takes one or two 16-bit words.
The size of the program memory is usually indicated in the naming of the device itself (e.g., the ATmega64x line has 64 kB of flash, while the ATmega32x line has 32 kB).
There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash. However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

Internal registers

The AVRs have 32 single-byte registers and are classified as 8-bit RISC devices. In the tinyAVR and megaAVR variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses (000016–001F16), followed by 64 I/O registers (002016–005F16). In devices with many peripherals, these registers are followed by 160 "extended I/O" registers, only accessible as memory-mapped I/O (006016–00FF16). Actual SRAM starts after these register sections, at address 006016 or, in devices with "extended I/O", at 010016. Even though there are separate addressing schemes and optimized opcodes for accessing the register file and the first 64 I/O registers, all can still be addressed and manipulated as if they were in SRAM.
The very smallest of the tinyAVR variants use a reduced architecture with only 16 registers (r0 through r15 are omitted) which are not addressable as memory locations. I/O memory begins at address 000016, followed by SRAM. In addition, these devices have slight deviations from the standard AVR instruction set. Most notably, the direct load/store instructions (LDS/STS) have been reduced from 2 words (32 bits) to 1 word (16 bits), limiting the total direct addressable memory (the sum of both I/O and SRAM) to 128 bytes. Conversely, the indirect load instruction's (LD) 16-bit address space is expanded to also include non-volatile memory such as Flash and configuration bits; therefore, the LPM instruction is unnecessary and omitted.
In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM. Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space. Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes (000016–0FFF16). As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions). Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space, which can be used optionally for mapping the internal EEPROM to the data address space (100016–1FFF16). The actual SRAM is located after these ranges, starting at 200016.

GPIO ports

Each GPIO port on a tiny or mega AVR drives up to eight pins and is controlled by three 8-bit registers: DDRx, PORTx and PINx, where x is the port identifier.
DDRx: Data Direction Register, configures the pins as either inputs or outputs.
PORTx: Output port register. Sets the output value on pins configured as outputs. Enables or disables the pull-up resistor on pins configured as inputs.
PINx: Input register, used to read an input signal. On some devices (but not all, check the datasheet), this register can be used for pin toggling: writing a logic .

GSM technology:

GSM refers to second-generation wireless telecommunications standard for digital cellular services. First deployed in Europe, it is based on TDMA (Time Division Multiple Access) technology. GSM uses three frequency bands: 900 MHz, 1800 MHz and 1900 MHz. Dual-band phones operate on two out of three of these frequencies, while tri-band phones operate on all three frequencies.
GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile),
It is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones. The GSM standard was developed as a replacement for first generation (1G) analog cellular networks, and originally described a digital, circuit switched network optimized for full duplex voice telephony. This was expanded over time to include data communications, first by circuit switched transport, then packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS).Further improvements were made when the 3GPP developed third generation (3G) UMTS standards followed by fourth generation (4G) LTE Advanced standards."GSM" is a trademark owned by the GSM Association.

IV. SOFTWARE

An embedded system is an application that contains at least one programmable computer (typically in the form of a microcontroller, a microprocessor or digital signal processor chip) and which is used by individuals who are, in the main, unaware that the system is computer-based.

Introduction

Looking around, we find ourselves to be surrounded by various types of embedded systems. Be it a digital camera or a mobile phone or a washing machine, all of them has some kind of processor functioning inside it. Associated with each processor is the embedded software. If hardware forms the body of an embedded system, embedded processor acts as the brain, and embedded software forms its soul. It is the embedded software which primarily governs the functioning of embedded systems.

During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check correct execution of the program. Some 'very fortunate' developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well.
As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

Initially C was developed by Kernighan and Ritchie to fit into the space of 8K and to write (portable) operating systems. Originally it was implemented on UNIX operating systems. As it was intended for operating systems development, it can manipulate memory addresses. Also, it allowed programmers to write very compact codes. This has given it the reputation as the language of choice for hackers too.

As assembly language programs are specific to a processor, assembly language didn't offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn't find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C. Subsequent sections will discuss what is Embedded C, features of C language, similarities and difference between C and embedded C, and features of embedded C programming.

EMBEDDED SYSTEMS PROGRAMMING

Embedded systems programming is different from developing applications on a desktop computers. Key characteristics of an embedded system, when compared to PCs, are as follows. Embedded devices have resource constraints(limited ROM, limited RAM, limited stack space, less processing power) Components used in embedded system and PCs are different; embedded systems typically uses smaller, less power consuming components. Embedded systems are more tied to the hardware.

Two salient features of Embedded Programming are code speed and code size. Code speed is governed by the processing power, timing constraints, whereas code size is governed by available program memory and use of programming language. Goal of embedded system programming is to get maximum features in minimum space and minimum time.

Embedded systems are programmed using different type of language

- Machine Code
- Low level language, i.e., assembly
- High level language like C, C++, Java, Ada, etc.
- Application level language like Visual Basic, scripts, Access, etc.

Assembly language maps mnemonic words with the binary machine codes that the processor uses to code the instructions. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high level languages are preferred for embedded systems programming.
Use of C in embedded systems is driven by following advantages it is small and reasonably simpler to learn, understand, program and debug. C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
• Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems. As C combines functionality of assembly language and features of high level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'. It is fairly efficient. It supports access to I/O and provides ease of management of large embedded projects.
Many of these advantages are offered by other languages also, but what sets C apart from others like Pascal, FORTRAN, etc. is the fact that it is a middle level language; it provides direct hardware control without sacrificing benefits of high level languages. Compared to other high level languages, C offers more flexibility because C is relatively small, structured language; it supports low-level bit-wise data manipulation.
Compared to assembly language, C Code written is more reliable and scalable, more portable between different platforms (with some changes). Moreover, programs developed in C are much easier to understand, maintain and debug. Also, as they can be developed more quickly, codes written in C offers better productivity. C is based on the philosophy 'programmers know what they are doing'; only the intentions are to be stated explicitly. It is easier to write good code in C & convert it to an efficient assembly code (using high quality compilers) rather than writing an efficient

code in assembly itself. Benefits of assembly language programming over C are negligible when we compare the ease with which C programs are developed by programmers.
Objected oriented language, C++ is not apt for developing efficient programs in resource constrained environments like embedded devices. Virtual functions & exception handling of C++ are some specific features that are not efficient in terms of space and speed in embedded systems. Sometimes C++ is used only with very few features, very much as C.

V.CONCLUSION

This paper describes a portable wireless Brain-Ma-chine-Brain Interface (BMBI) that links the brain to external hardware, In conclusion, we will be able to transfer ourselves into computers at some point. Most arguments against this outcome are seemingly easy to circumvent. They are either simple minded, or simply require further time for technology to increase. The only serious threats raised are also overcome as we note the combination of biological and digital technologies. They will be able to provide an interface with computers that is as close as our mind can be while we still reside in our biological form. Nanobots could also carefully scan the structure of our brain, providing a complete readout of the connections between each neuron. They would also record the current state of the brain. This information, when entered into a computer, could then continue to function as us. All that is required is a computer with large enough storage space and processing power.