CHAPTER 1
INTRODUCTION

Braille is the main system of conveying information by tactile sensation, and it was designed for visually impaired people. There are more than 200 million visually impaired people in the world, and this number is on the rise [. Braille is easily accessible in everyday life, but mainly provides information for guidance in public facilities and cannot provide personalized information for individuals. Meanwhile, as technology advances rapidly, it has become possible to access desired information anytime and anywhere through portable personal devices such as smartphones, but it mainly provides information through a visual display. Therefore, the development of the braille device is essential to reduce the gap of information acquisition between visually impaired people and non-disabled people. A refreshable braille display solves the intrinsic problems of braille that it cannot reproduce different information and is bulky. Various principles have been applied to the refreshing mechanism but the most universal principle is piezo-actuator based braille display modules. These actuators automatically raise and lower braille pins, but are limited to displaying braille information. A braille cell is composed of 6 to 8 pins and represents a single character, and the pin-to-pin and cell-to-cell spacing are different, which means that there is a limit to expression of information other than text information. In this regard, research on tactile display has been conducted to provide two-dimensional (2D) information through an even pin-to-pin spacing of array type display. If 2D information is provided, not only text, but also graphical information such as pictures, maps, and plots can be expressed [11], [12], and more interactive expression can be represented [13]. This will help visually impaired people in education, mobility, and life convenience, and non-disabled people will also be able to use it in environments with limited vision. In order to satisfy the small pin-to-pin spacing for all pins, the actuator must be much smaller, and at the same time, they must be able to produce a protruding force that can be felt through our fingers.

Tactile displays are also being researched based on various operating principles, and we propose an actuator based on an electromagnetic (EM) principle. An EM actuator is composed of the voice coil and permanent magnet and utilizes their attractive/repulsive force. It is mainly applied to the vibration motor of portable devices due to its low operating voltage and ease of miniaturization [14]. Since refreshing braille has a relatively simple mechanism, requiring only linear motion (up and down), applying EM actuators to braille display seems appropriate. However, application to a tactile display requires several challenges. First of all, an EM actuator moves the magnet through the magnetic force, which means that the magnetic interference with the surroundings must be considered. In order for the finger to detect the curvature while moving on the display, the braille pin must maintain a certain amount of protruding force, especially if intended to remain as a fixed statement for some duration. On the other hand, the generated magnetic field must be weak so as not to affect the surrounding pins during driving. This will have a trade-off relationship and must be resolved through the design of its magnetic structure. Second, the issue of heat generation should be considered.

The tactile display consumes energy not only when driving the pins, but also when maintaining the positions of pins. Since the continuous flow of current on the solenoid generates heat, heat dissipation was an essential condition for the design of the EM actuator. Finally, the energy consumption should be considered. Since the tactile display is composed of a large array of pins, the energy consumption per each pin must be low. In particular, to be applied to portable devices, the energy consumption rate needs to meet the power and capacity limits of its battery.

EMBEDDED SYSTEM

Embedded system is a combination of hardware and software used to achieve a single task. Embedded system is computer systems that monitor, respond to, or control an external environment. Environment connected to system through sensors, actuators and other I/O devices. Embedded system must meet timing and other constrains imposed on it by environment.

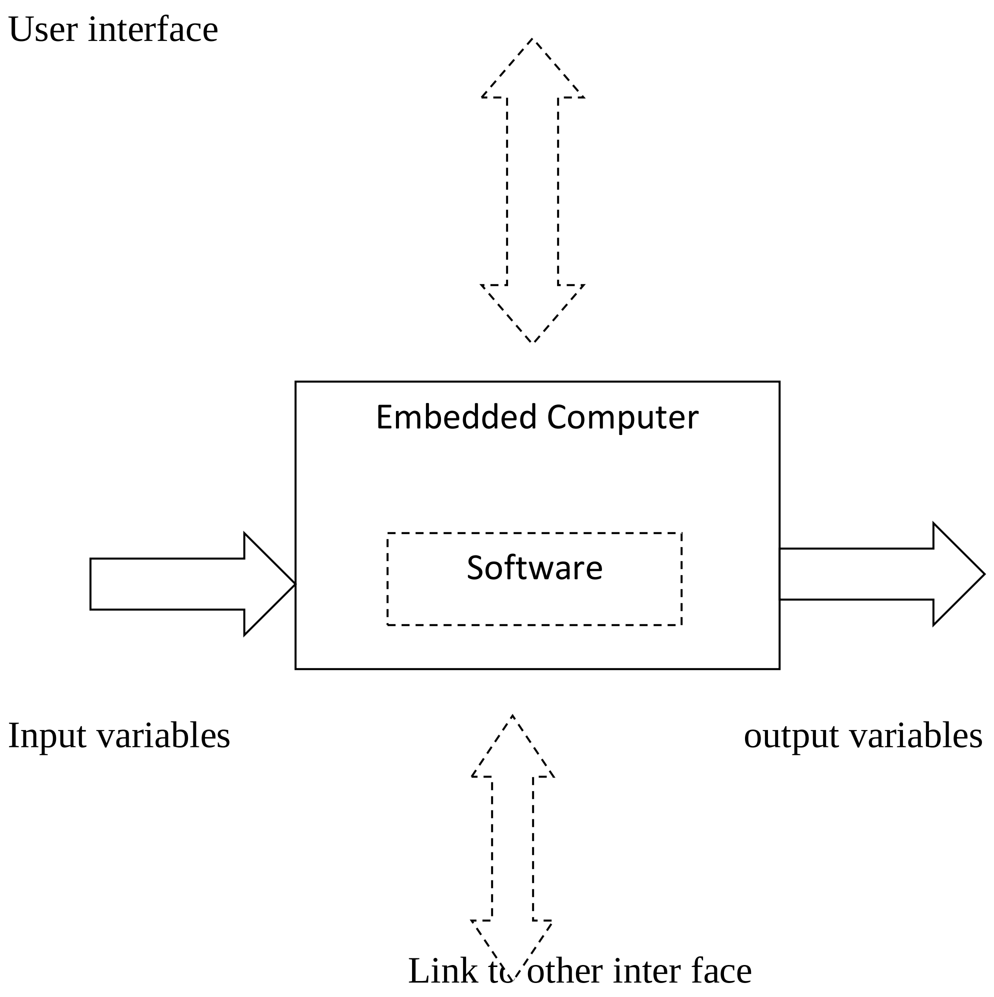


Fig.1.1 Blocks of Embedded System

Variety of embedded systems

An embedded RouterBoard 112 with U.FL-RSMA pigtail and R52 miniPCI Wi-Fi card widely used by wireless Internet service providers (WISPs) in the Czech Republic.Embedded systems span all aspects of modern life and there are many examples of their use.Telecommunications systems employ numerous embedded systems from telephone switches for the network to mobile phones at the end-user. Computer networking uses dedicated routers and network bridges to route data.

Consumer electronics include personal digital assistants (PDAs), mp3 players, mobile phones, videogame consoles, digital cameras, DVD players, GPS receivers, and printers. Many household appliances, such as microwave ovens, washing machines and dishwashers, are including embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors — brushless DC motors, induction motors and DC motors — are using electric/electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles are increasingly using embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.Medical equipment is continuing to advance with more embedded systems for vital signs monitoring, electronic stethoscopes for amplifying sounds, and various medical imaging (PET, SPECT, CT, MRI) for non-invasive internal inspections.

Characteristics

1. Embedded systems are designed to do some specific task, rather than be a general-purpose computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.

2. Embedded systems are not always standalone devices. Many embedded systems consist of small, computerized parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music. Similarly, an embedded system in an automobile provides a specific function as a subsystem of the car itself.

3. The program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or Flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard or screen.

User interface

Embedded systems range from no user interface at all — dedicated only to one task — to complex graphical user interfaces that resemble modern computer desktop operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (for example popular HD44780 LCD) with a simple menu system.

More sophisticated devices which use a graphical screen with touch sensing or screen-edge buttons provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behavior of pointing at what's desired. Handheld systems often have a screen with a "joystick button" for a pointing device.

Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I²C, etc.) or network (e.g. Ethernet) connection. This approach gives several advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies BSP, allows us to build rich user interface on the PC. A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network routers. The user interface is displayed in a web browser on a PC connected to the device, therefore needing no bespoke software to be installed.

Processors in embedded systems

Secondly, embedded processors can be broken into two broad categories: ordinary microprocessors (μP) and microcontrollers (μC), which have many more peripherals on chip, reducing cost and size. Contrasting to the personal computer and server markets, a fairly large number of basic CPU architectures are used; there are Von Neumann as well as various degrees of Harvard architectures, RISC as well as non-RISC and VLIW; word lengths vary from 4-bit to 64-bits and beyond (mainly in DSP processors) although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

Readymade computer boards

PC/104 and PC/104+ are examples of standards for readymade computer boards intended for small, low-volume embedded and ruggedized systems, mostly x86-based. These are often physically small compared to a standard PC, although still quite large compared to most simple (8/16-bit) embedded systems. They often use MSDOS, Linux, NetBSD, or an embedded real-time operating system such as MicroC/OS-II, QNX or VxWorks. Sometimes these boards use non-x86 processors.

In certain applications, where small size or power efficiency are not primary concerns, the components used may be compatible with those used in general purpose x86 personal computers. Boards such as the VIA EPIA range help to bridge the gap by being PC-compatible but highly integrated, physically smaller or have other attributes making them attractive to embedded engineers. The advantage of this approach is that low-cost commodity components may be used along with the same software development tools used for general software development. Systems built in this way are still regarded as embedded since they are integrated into larger devices and fulfill a single role. Examples of devices that may adopt this approach are ATMs and arcade machines, which contain code specific to the application.

However, most ready-made embedded systems boards are not PC-centered and do not use the ISA or PCI busses. When a System-on-a-chip processor is involved, there may be little benefit to having a standarized bus connecting discrete components, and the environment for both hardware and software tools may be very different.

One common design style uses a small system module, perhaps the size of a business card, holding high density BGA chips such as an ARM-based System-on-a-chip processor and peripherals, external flash memory for storage, and DRAM for runtime memory. The module vendor will usually provide boot software and make sure there is a selection of operating systems, usually including Linux and some real time choices. These modules can be manufactured in high volume, by organizations familiar with their specialized testing issues, and combined with much lower volume custom mainboards with application-specific external peripherals. Gumstix product lines are a Linux-centric example of this model.

ASIC and FPGA solutions

A common array of n configuration for very-high-volume embedded systems is the system on a chip (SoC) which contains a complete system consisting of multiple processors, multipliers, caches and interfaces on a single chip. SoCs can be implemented as an application-specific integrated circuit (ASIC) or using a field-programmable gate array (FPGA).

Peripherals

Embedded Systems talk with the outside world via peripherals, such as:

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485 etc.
- Synchronous Serial Communication Interface: I2C, SPI, SSC and ESSI (Enhanced Synchronous Serial Interface)
- Universal Serial Bus (USB)
- Multi Media Cards (SD Cards, Compact Flash etc.)
- Networks: Ethernet, LonWorks, etc.
- Fieldbuses: CAN-Bus, LIN-Bus, PROFIBUS, etc.
- Timers: PLL(s), Capture/Compare and Time Processing Units
- Discrete IO: aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging: JTAG, ISP, ICSP, BDM Port, BITP, and DP9 ports.

Reliability

Embedded systems often reside in machines that are expected to run continuously for years without errors, and in some cases recover by themselves if an error occurs. Therefore the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

1. The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
2. The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals.
3. The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and also soft errors in the hardware:

- watchdog timer that resets the computer unless the software periodically notifies the watchdog
- subsystems with redundant spares that can be switched over to
- software "limp modes" that provide partial function
- Designing with a Trusted Computing Base (TCB) architecture ensures a highly secure & reliable system environment
- An Embedded Hypervisor is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This

encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.

- Immunity Aware Programming

CHAPTER 2

SYSTEM DESIGN

EXISTING SYSTEM

Existing System is an includes a smart glove that translates the Braille alphabet, which is used almost universally by the literate deaf blind population, into text and vice versa, The wearer can perceive and interpret incoming messages by tactile feedback patterns of mini vibrational motors on the dorsal side of the glove. The successful implementation of real-time two- way translation between English and Braille, and communication of the wearable device with a mobile phone/PC opens up new opportunities of information exchange which were hitherto un- available to deaf blind individuals, such as remote communication, as well as parallel one-to many broadcast. The glove also makes communicating with laypersons without knowledge of Braille possible, without the need for trained interpreters. A Braille writer through which the deaf blind and the blind persons can write the Braille script.

DISADVANTAGES OF EXISTING SYSTEM

- Requirements Of High-Voltage Supplies.
- High Price.
- Low Flexibility.
- Low Accuracy.

PROPOSED SYSTEM:

The Proposed System help the blind people to communicates the message via mobile android application. It enables user to convey simple messages by capacitive touch sensors as input sensors placed on the Palmer side of the Message converted to text by the mobile phone.

But here, using this Braille system both reading and replying the messages possible by visually impaired people. This system helps the blind people can access the message application in mobiles as a normal people. At the same time, the keypad is used to send one by one message based on keypad options so the user can send the information through messages easily. This system is also applicable for uneducated people.

ADVANTAGES OF PROPOSED SYSTEM:

- Fast Refresh Rate.
- Low Cost.
- Portable.
- Small Size.
- Easy Access.

APPLICATIONS OF PROPOSED SYSTEM:

- Blind People System.
- Easy Message Alert System.

BLOCK DIAGRAM:

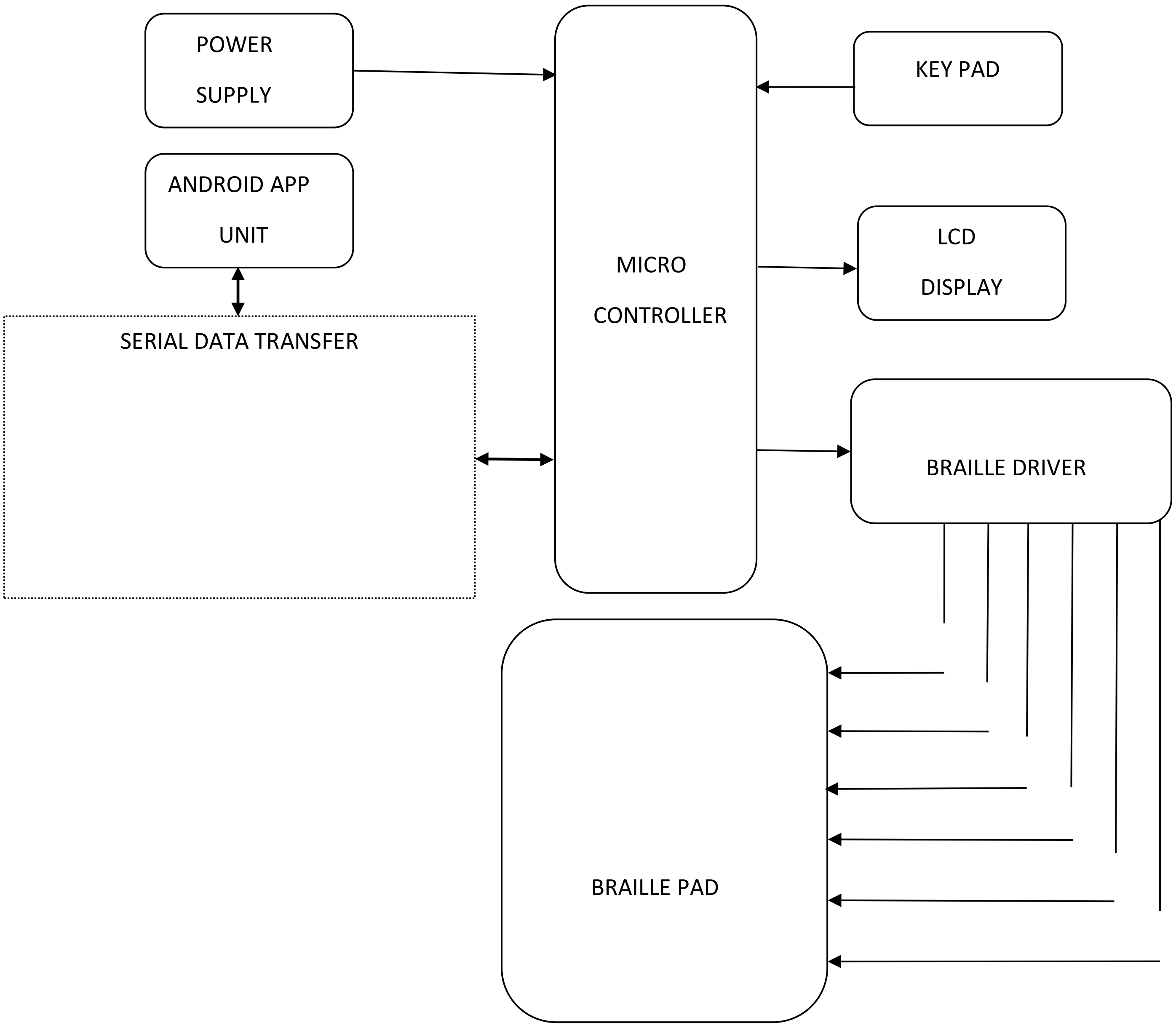


Fig 1. Braille-Based Mobile Communication Model

HARDWARE REQUIREMENTS:

- MICRO CONTROLLER
- LCD DISPLAY
- KEY PAD
- SERIAL DATA TRANSFER
- BRAILLE DRIVER
- BRAILLE PAD
- BRAILLE MOTORS (6)
- BLUETOOTH

SOFTWARE REQUIREMENTS:

- EMBEDDED C
- ARDUINO IDE

CHAPTER 3

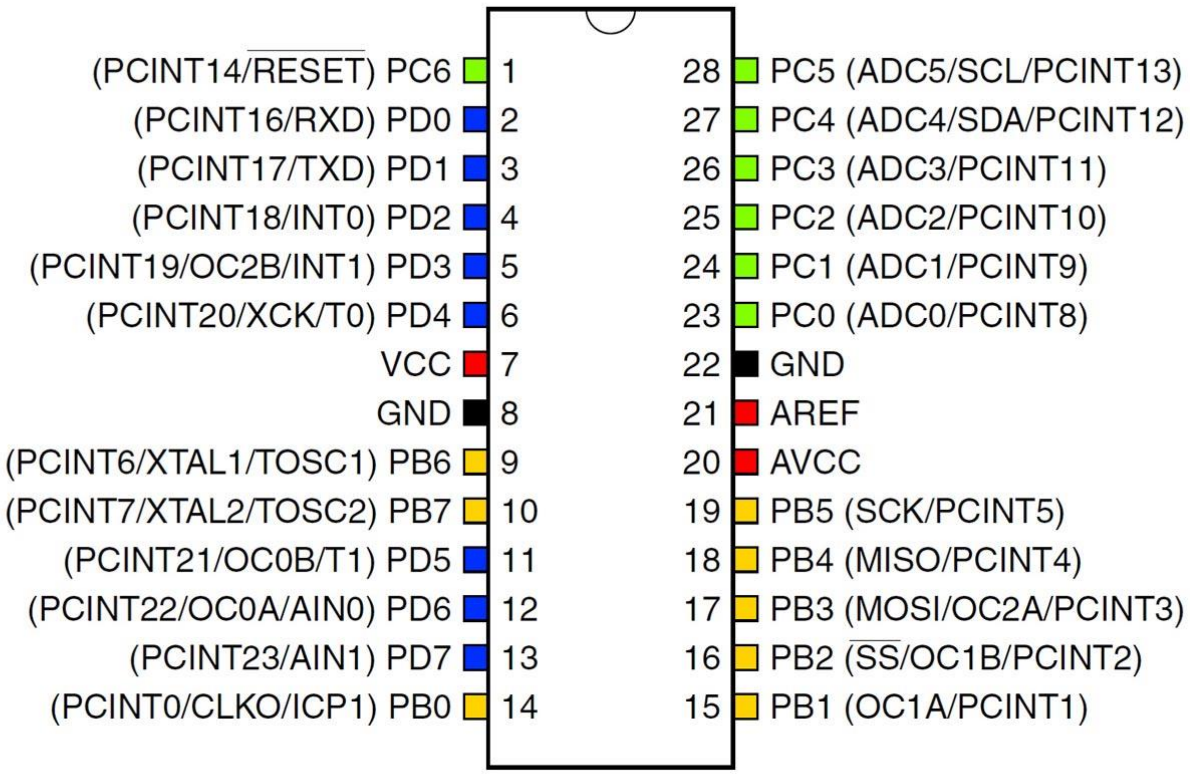
CONTROLLER

ARDUINO:

Arduino ATMEGA-328 microcontroller consist of 14 input and output analog and digital pins (from this 6 pins are considered to be a PWM pins), 6 analog inputs and remaining digital inputs. Power jack cable is used to connect arduino board with the computer. Externally battery is connected with the arduino microcontroller for the power supply. Arduino is an open source microcontroller from which there is no feedback present in the microcontroller. This arduino board consist of I2C bus that can be able to transfer the data from arduino board to the output devices. These arduino boards are programmed over RS232 serial interface connections with ATmega arduino microcontrollers. The operating volt ranges from 5v. The input voltage recommended for arduino microcontroller is from 7v and the maximum of 12v. The DC input current given to the arduino board is in the range of 40mA.

It consists of different types of memories such as flash memory, EEPROM, SRAM. The length of the arduino board is nearly about 68.64mm and the width of the microcontroller is about 53.4mm. The weight of the arduino microcontroller is about 20g. We can use various types of microcontroller such as 8 bit AVL Atmel microcontroller and 32 bit Atmel arm microprocessor. From these different kinds of processors, we can use those processors for various engineering projects as well as industrial applications. Some of the examples of using the arduino in the industrial applications are controlling the actuators and sensors. Some of the examples of arduino microcontrollers are Arduino Duemilanove, Arduino UNO, Arduino Leonardo, Arduino Mega, and Arduino MEGA 2560 R3, Arduino MEGA 2560 R3, Arduino Nano, Arduino Due, LilyPadArduino, micro arduino. We have already mentioned, arduino has been programmed by using c and c++ programming language. These c and c++ are high level languages. Normally it has 18 number of input and output pins. Among those 6 pins are considered to be an analog inputs.From these analog inputs, we can be able to work the arduino microcontroller using analog inputs supply. Normally analog inputs can be in the range of 0-5V. Similar to that digital inputs are present in the microcontroller which can act the use of microcontroller using digital inputs. Digital inputs can be in the range of 5V.

PIN DIAGRAM



ATMEGA 328 microcontroller, which acts as a processor for the arduino board. Nearly it consists of 28 pins. From these 28 pins, the inputs can be controlled by transmitting and receiving the inputs to the external device. It also consists of pulse width modulation (PWM). These PWM are used to transmit the entire signal in a pulse modulation. Input power supply such as Vcc and Gnd are used. These IC mainly consists of analog and digital inputs. These analog and digital inputs are used for the process of certain applications.

The power pins are as follows:

- VIN: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V: The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND: Ground pins.

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX): Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- 2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library.
- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Analog Input:

Arduino atmega-328 microcontroller board consist of 6 analog inputs pins. These analog inputs can be named from A0 to A5. From these 6 analog inputs pins, we can do the process by using analog inputs. Analog inputs can be used in the operating range of 0 to 5V. Analog signal is considered as the continuous time signal, from which these analog signal can be used for certain applications. These are also called as non-discrete time signal. Inputs such as voltage, current etc.., are considered to beeither analog signal or digital signal only by analysing thetime signal properties. Various applications of arduinomicrocontroller can use only an analog input instead of digital inputs. For these applications, analog input ports or pins can be used.

Digital Input:

Digital inputs can be defined as the non-continuous time signal with discrete input pulses. It can be represented as 0's and 1's. These digital inputs can be either on state or in off state. Arduino atmega328 microcontroller also consists of 12 digital input pins. It can be stated as D0 to D11. Nearly 12 inputs can be used for digital input/output applications. The working of the digital input ports is where the discrete input pulses can be triggered and supplied to the ports. These ports receive the input and therefore the port can be used for both input and output process. These digital pins can access only the digital inputs.

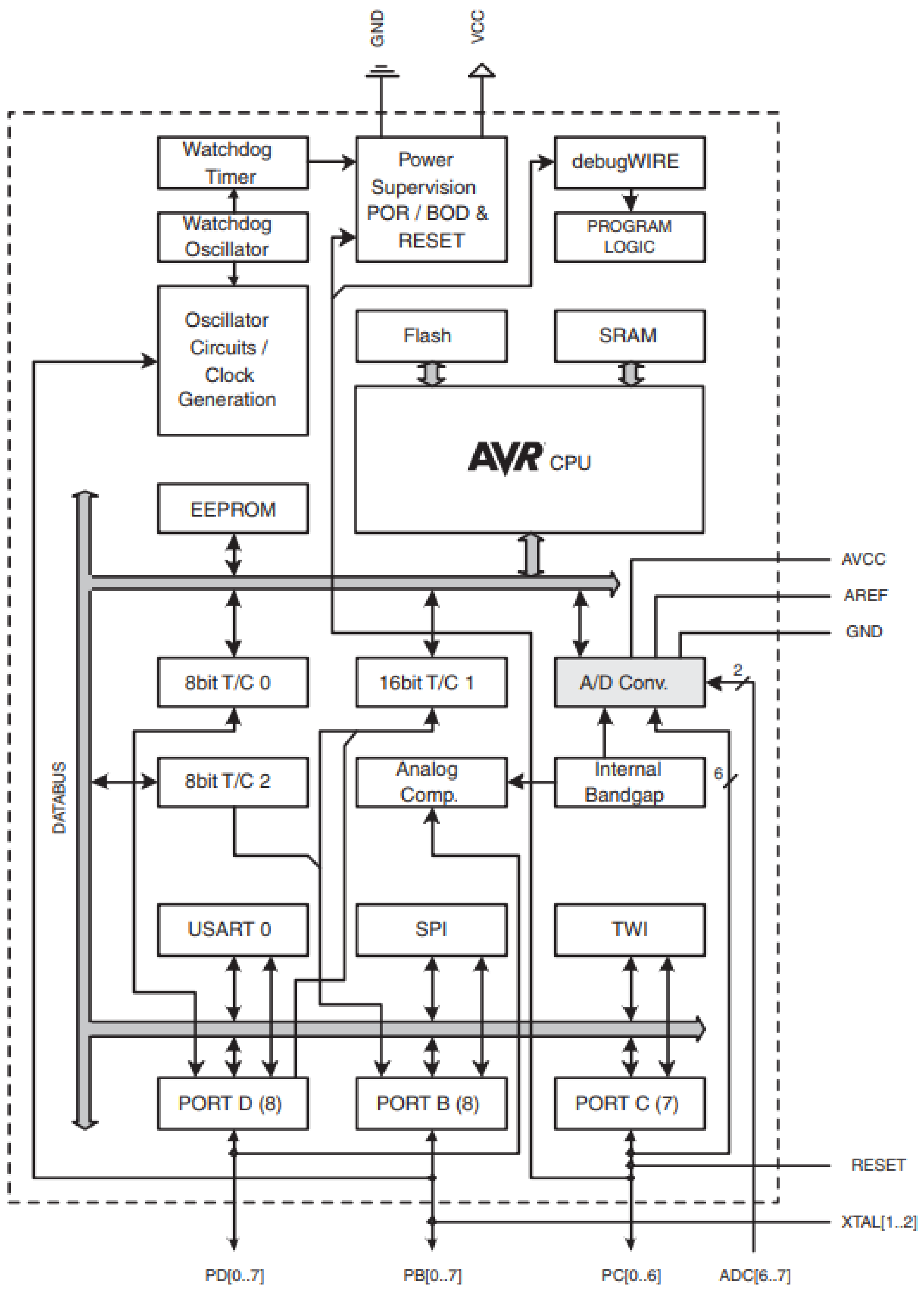
Power Jack Cable / USB PORT:

This Arduino atmega-328 microcontroller can be interfaced with the other electronic devices such as computer by using USB port or power jack cable from these power jack cable, we can upload the program of Arduino for their applications. At first, the program can be initialised or can be edited by using Arduino software tools. Then these programs can be transferred through arduino microcontroller board by using usb cable or power jack cable.

Power Supply:

There is an additional power supply source present in Arduino microcontroller. Power supply port is present at the corner of the arduino microcontroller. Either we can use this power supply port by connecting with external power supply.(ie, ac power supply), or by connecting an dc power supply through input pins. These power supplies produce an active form to the arduino microcontroller. These arduino microcontrollers can accept a range of power supply. When the power supply voltage rangeexceeds, the microcontroller gets damaged. Hence, onlythe particular range of power supply should be given to thearduino microcontroller.

Architecture Diagram:



The AVR core combines a rich instruction set with 32 general purpose working registers. All the32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independentregisters to be accessed in one single instruction executed in one clock cycle. The resultingarchitecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The ATmega48PA/88PA/168PA/328P provides the following features: 4/8/16/32K bytes of In System Programmable Flash with Read-While-Write capabilities, 256/512/512/1K bytes EEPROM, 512/1K/1K/2K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section willcontinue to run while the Application Flash section is updated, providing true Read-While-Writeoperation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on amonolithic chip, the Atmel ATmega48PA/88PA/168PA/328P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

Communication:

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

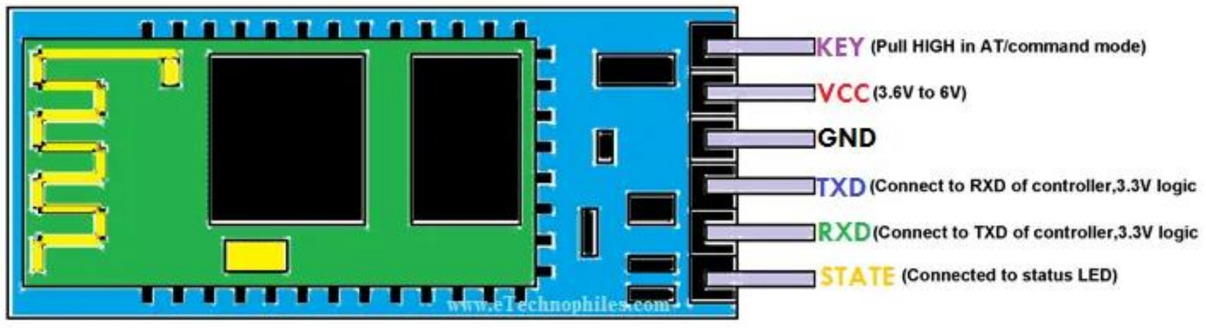
CHAPTER 4

BLUETOOTH

BLUETOOTH

A Bluetooth technology is a high speed low powered wireless technology link that is designed to connect phones or other portable equipment together. It is a specification (IEEE 802.15.1) for the use of low power radio communications to link phones, computers and other network devices over short distance without wires. Wireless signals transmitted with Bluetooth cover short distances, typically up to 30 feet (10 meters).

It is achieved by embedded low cost transceivers into the devices. It supports on the frequency band of 2.45GHz and can support upto 721KBps along with three voice channels. This frequency band has been set aside by international agreement for the use of industrial, scientific and medical devices (ISM).rd-compatible with 1.0 devices.


KEY (Pull HIGH in AT/command mode)
VCC (3.6V to 6V)
GND
TXD (Connect to RXD of controller,3.3V logic
RXD (Connect to TXD of controller,3.3V logic
STATE (Connected to status LED)
www.eTechnophiles.com

Bluetooth can connect up to “eight devices” simultaneously and each device offers a unique 48 bit address from the IEEE 802 standard with the connections being made point to point or multipoint.

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication. This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

Bluetooth Module HC-05

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc.

Hardware Features

- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- 3.3 to 5 V I/O.
- PIO(Programmable Input/Output) control.
- UART interface with programmable baud rate.
- With integrated antenna.
- With edge connector.

Pin Description

The HC-05 Bluetooth Module has 6pins. They are as follows:

ENABLE: When enable is pulled LOW, the module is disabled which means the module will not turn on and it fails to communicate. When enable is left open or connected to 3.3V, the module is enabled i.e the module remains on and communication also takes place.

Vcc: Supply Voltage 3.3V to 5V

GND: Ground pin

TXD & RXD: These two pins acts as an UART interface for communication

STATE: It acts as a status indicator. When the module is not connected to / paired with any other Bluetooth device, signal goes Low. At this low state, the led flashes continuously which denotes that the module is not paired with other device. When this module is connected to/paired with any other Bluetooth device, the signal goes High. At this high state, the led blinks with a constant delay say for example 2s delay which indicates that the module is paired.

BUTTON SWITCH: This is used to switch the module into AT command mode. To enable AT command mode, press the button switch for a second. With the help of AT commands, the user can change the parameters of this module but only when the module is not paired with any other BT device. If the module is connected to any other Bluetooth device, it starts to communicate with that device and fails to work in AT command mode.

CHAPTER 5

LCD

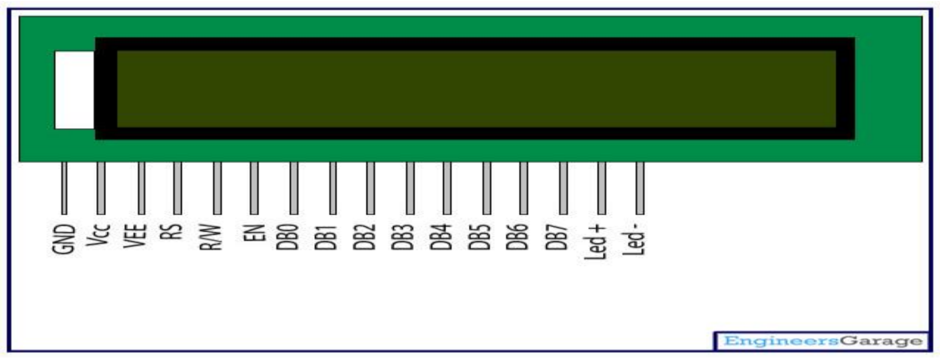
LIQUID CRYSTAL DISPLAY

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is a very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

Pin Diagram:



Pin Description:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	

INTRODUCTION:

Serial communication is basically the transmission or reception of data one bit at a time. Today's computers generally address data in bytes or some multiple thereof. A byte contains 8 bits. A bit is basically either a logical 1 or zero. Every character on this page is actually expressed internally as one byte. The serial port is used to convert each byte to a stream of ones and zeroes as well as to convert a stream of ones and zeroes to bytes. The serial port contains a electronic chip called a Universal Asynchronous Receiver/Transmitter (UART) that actually does the conversion.

The baud rate is the number of times the signal can switch states in one second. Therefore, if the line is operating at 9600 baud, the line can switch states 9,600 times per second.

Half duplex serial communication needs at a minimum two wires, signal ground and the data line. Full duplex serial communication needs at a minimum three wires, signal ground, transmit data line, and receive data line. The RS232 specification governs the physical and electrical characteristics of serial communications. This specification defines several additional signals that are asserted (set to logical 1) for information and control beyond the data signal

These signals are the Carrier Detect Signal (CD), asserted by modems to signal a successful connection to another modem, Ring Indicator (RI), asserted by modems to signal the phone ringing, Data Set Ready (DSR), asserted by modems to show their presence, Clear To Send (CTS), asserted by modems if they can receive data, Data Terminal Ready (DTR), asserted by terminals to show their presence, Request To Send (RTS), asserted by terminals if they can receive data. The section RS232 Cabling describes these signals and how they are connected.

NULL MODEM:

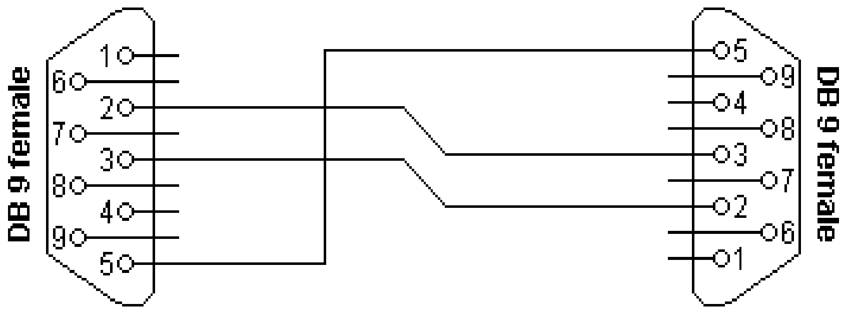
Serial communications with RS232. One of the oldest and most widely spread communication methods in computer world. The way this type of communication can be performed is pretty well defined in standards. I.e. with one exception. The standards show the use of DTE/DCE communication, the way a computer should communicate with a peripheral device like a modem. For your information, DTE means *Data Terminal Equipment* (computers etc.) where DCE is the abbreviation of *Data Communication Equipment* (modems).

RS232:

When we look at the connector pin out of the RS232 port, we see two pins which are certainly used for flow control. These two pins are RTS, request to send and CTS, clear to send. With DTE/DCE communication (i.e. a computer communicating with a modem device) RTS is an output on the DTE and input on the DCE. CTS are the answering signal coming from the DCE.

NULL MODEM WITHOUT HANDSHAKING:

How to use the handshaking lines in a null modem configuration? The simplest way is to don't use them at all. In that situation, only the data lines and signal ground are cross connected in the null modem communication cable. All other pins have no connection. An example of such a null modem cable without handshaking can be seen in the figure below.



Connector 1	Connector 2	Function
2	3	Rx $\leftarrow$ TX
3	2	TX $\rightarrow$ Rx
5	5	Signal ground

Fig.4.1. Simple null modem without handshaking

COMPATIBILITY ISSUES:

If you read about null modems, this three wire null modem cable is often talked about. Yes, it is simple but can we use it in all circumstances? There is a problem, if either of the two devices checks the DSR or CD inputs. These signals normally define the ability of the other side to communicate. As they are not connected, their signal level will never go high. This might cause a problem.

The same holds for the RTS/CTS handshaking sequence. If the software on both sides is well structured, the RTS output is set high and then a waiting cycle is started until a ready signal is received on the CTS line. This causes the software to hang because no physical connection is present to either CTS line to make this possible. The only type of communication which is allowed on such a null modem line is data-only traffic on the cross connected Rx/TX lines.

KEYPAD

GENERAL DESCRIPTION

A miniature keyboard or set of buttons for operating a portable electronic device, telephone, or other equipment it is also possible to use numerous additional modules linked to the development system through the I/O port connectors. Some of these additional modules can operate as stand-alone devices without being connected to the microcontroller.

PRODUCT DESCRIPTION

A keypad is a set of buttons arranged in a block or pad for a specific task. It contains 5 keys arranged in matrix format. The pulses from the microcontroller are used for switching keys in a keypad. In order the keypad to work properly, pull-down resistors should be placed on the microcontroller’s input pins, thus defining logic state when no button is pressed.

By combining zeros and ones on the output pins, it is determined which button is pressed. It does not require separate power supply for switching. The keypad may be used for a multi input switching.

5x1 Keypad

FEATURES

- No need of input voltage
- Multi input switching

APPLICATIONS

- Switching applications
- Speed control application

CHAPTER 6
SOFTWARE DETAILS

Arduino Software (IDE):

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

Writing Sketches:

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the Arduino Software (IDE) prior to 1.0 saved sketches with the extension .pde. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the .ino extension on save.

Verify: Checks your code for errors compiling it.

Upload: Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New: Creates a new sketch.

Open: Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn’t scroll; if you need to open a sketch late in the list, use the File | Sketchbook menu instead.

Save: Saves your sketch.

Serial Monitor: Opens the serial monitor.

Additional commands are found within the five menus: File, Edit, Sketch, Tools, and Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

File:

New: Creates a new instance of the editor, with the bare minimum structure of a sketch already in place.

Open: Allows to load a sketch file browsing through the computer drives and folders.

Open Recent: Provides a short list of the most recent sketches, ready to be opened.

Sketchbook: Shows the current sketches within the sketchbook folder structure; clicking on any name opens the corresponding sketch in a new editor instance.

Examples: Any example provided by the Arduino Software (IDE) or library shows up in this menu item. All the examples are structured in a tree that allows easy access by topic or library.

Close: Closes the instance of the Arduino Software from which it is clicked.

Save: Saves the sketch with the current name. If the file hasn't been named before, a name will be provided in a "Save as.." window.

Save as: Allows to save the current sketch with a different name.

Page Setup: It shows the Page Setup window for printing.

Print: Sends the current sketch to the printer according to the settings defined in Page Setup.

Preferences: Opens the Preferences window where some settings of the IDE may be customized, as the language of the IDE interface.

Quit: Closes all IDE windows. The same sketches open when Quit was chosen will be automatically reopened the next time you start the IDE.

Edit:

Undo/Redo: Goes back of one or more steps you did while editing; when you go back, you may go forward with Redo.

Cut: Removes the selected text from the editor and places it into the clipboard.

Copy: Duplicates the selected text in the editor and places it into the clipboard.

Copy for Forum: Copies the code of your sketch to the clipboard in a form suitable for posting to the forum, complete with syntax coloring.

Copy as HTML: Copies the code of your sketch to the clipboard as HTML, suitable for embedding in web pages.

Paste: Puts the contents of the clipboard at the cursor position, in the editor.

Select All

Selects and highlights the whole content of the editor.

Comment/Uncomment

Puts or removes the // comment marker at the beginning of each selected line.

Increase/Decrease Indent

Adds or subtracts a space at the beginning of each selected line, moving the text one space on the right or eliminating a space at the beginning.

Find

Opens the Find and Replace window where you can specify text to search inside the current sketch according to several options.

Find Next

Highlights the next occurrence - if any - of the string specified as the search item in the Find window, relative to the cursor position.

Find Previous

Highlights the previous occurrence - if any - of the string specified as the search item in the Find window relative to the cursor position.

Sketch

Verify/Compile: Checks your sketch for errors compiling it; it will report memory usage for code and variables in the console area.

Upload: Compiles and loads the binary file onto the configured board through the configured Port.

Upload Using Programmer: This will overwrite the bootloader on the board; you will need to use Tools > Burn Bootloader to restore it and be able to Upload to USB serial port again. However, it allows you to use the full capacity of the Flash memory for your sketch. Please note that this command will NOT burn the fuses. To do so a Tools -> Burn Bootloader command must be executed.

Export Compiled Binary: Saves a .hex file that may be kept as archive or sent to the board using other tools.

Show Sketch Folder: Opens the current sketch folder.

Include Library: Add a library to your sketch by inserting #include statements at the start of your code. For more details, see libraries below. Additionally, from this menu item you can access the Library Manager and import new libraries from .zip files.

Add File: Adds a source file to the sketch (it will be copied from its current location). The new file appears in a new tab in the sketch window. Files can be removed from the sketch using the tab menu accessible clicking on the small triangle icon below the serial monitor one on the right side o the toolbar.

Tools

Auto Format: This formats your code nicely: i.e. indents it so that opening and closing curly braces line up, and that the statements inside curly braces are indented more.

Archive Sketch: Archives a copy of the current sketch in .zip format. The archive is placed in the same directory as the sketch.

Fix Encoding & Reload: Fixes possible discrepancies between the editor char map encoding and other operating systems char maps.

Serial Monitor: Opens the serial monitor window and initiates the exchange of data with any connected board on the currently selected Port. This usually resets the board, if the board supports Reset over serial port opening.

Board: Select the board that you're using. See below for descriptions of the various boards.

Port: This menu contains all the serial devices (real or virtual) on your machine. It should automatically refresh every time you open the top-level tools menu.

Programmer: For selecting a harware programmer when programming a board or chip and not using the onboard USB-serial connection. Normally you won't need this, but if you're burning a bootloader to a new microcontroller, you will use this.

Burn Bootloader: The items in this menu allow you to burn a bootloader onto the microcontroller on an Arduino board. This is not required for normal use of an Arduino or Genuino board but is useful if you purchase a new ATmega microcontroller (which normally come without a bootloader). Ensure that you've selected the correct board from the Boards menu before burning the bootloader on the target board. This command also set the right fuses.

Help

Here you find easy access to a number of documents that come with the Arduino Software (IDE). You have access to Getting Started, Reference, this guide to the

IDE and other documents locally, without an internet connection. The documents are a local copy of the online ones and may link back to our online website.

Find in Reference: This is the only interactive function of the Help menu: it directly selects the relevant page in the local copy of the Reference for the function or command under the cursor.

Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook. You can view or change the location of the sketchbook location from with the Preferences dialog.

Tabs, Multiple Files, and Compilation

Allows you to manage sketches with more than one file (each of which appears in its own tab). These can be normal Arduino code files (no visible extension), C files (.c extension), C++ files (.cpp), or header files (.h).

Uploading: Before uploading your sketch, you need to select the correct items from the Tools > Board and Tools > Port menus. The boards are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an Uno or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, you look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx ,

/dev/ttyUSBx or similar. Once you've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, you'll need to press the reset button on the board just before starting the upload. On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete, or show an error.

When you upload a sketch, you're using the Arduino bootloader, a small program that has been loaded on to the microcontroller on your board. It allows you to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

Libraries:

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. If a sketch no longer needs a library, simply delete its #include statements from the top of your code.

There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch. See these instructions for installing a third-party library.

Third-Party Hardware

Support for third-party hardware can be added to the hardware directory of your sketchbook directory. Platforms installed there may include board definitions (which appear in the board menu), core libraries, bootloaders, and programmer definitions. To install, create the hardware directory, then unzip the third-party platform into its own sub-directory. (Don't use "arduino" as the sub-directory name or you'll override the built-in Arduino platform.) To uninstall, simply delete its directory.

Serial Monitor

This displays serial sent from the Arduino or Genuino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in your sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun your sketch) when you connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if your sketch needs a complete management of the serial communication with control characters, you can use an external terminal program and connect it to the COM port assigned to your Arduino board.

You can also talk to the board from Processing, Flash, MaxMSP, etc (see the interfacing page for details).

Boards

The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) used when compiling and uploading sketches; and sets and the file and fuse settings used by the burn bootloader command. Some of the board definitions differ only in the latter, so even if you've been uploading successfully with a particular selection you'll want to check it before burning the bootloader. You can find a comparison table between the various boards here.

Arduino Software (IDE) includes the built in support for the boards in the following list, all based on the AVR Core. The Boards Manager included in the standard installation allows to add support for the growing number of new boards based on different cores like Arduino Due, Arduino Zero, Edison, and Galileo and so on.

- **Arduino Yùn**

An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.

- **Arduino/Genuino Uno**

An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino Diecimila or Duemilanove w/ ATmega168**

An ATmega168 running at 16 MHz with auto-reset.

- **Arduino Nano w/ ATmega328P**

An ATmega328P running at 16 MHz with auto-reset. Has eight analog inputs.

- **Arduino/Genuino Mega 2560**

An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- **Arduino Mega**

An ATmega1280 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- **Arduino Mega ADK**

An ATmega2560 running at 16 MHz with auto-reset, 16 Analog In, 54 Digital I/O and 15 PWM.

- **Arduino Leonardo**

An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.

- **Arduino/Genuino Micro**

An ATmega32u4 running at 16 MHz with auto-reset, 12 Analog In, 20 Digital I/O and 7 PWM.

- **Arduino Esplora**

An ATmega32u4 running at 16 MHz with auto-reset.

- **Arduino Mini w/ ATmega328P**

An ATmega328P running at 16 MHz with auto-reset, 8 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino Ethernet**

Equivalent to Arduino UNO with an Ethernet shield: An ATmega328P running at 16 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino Fio**

An ATmega328P running at 8 MHz with auto-reset. Equivalent to Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328P, 6 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino BT w/ ATmega328P**

ATmega328P running at 16 MHz. The bootloader burned (4 KB) includes codes to initialize the on-board bluetooth module, 6 Analog In, 14 Digital I/O and 6 PWM.

- **LilyPad Arduino USB**

An ATmega32u4 running at 8 MHz with auto-reset, 4 Analog In, 9 Digital I/O and 4 PWM.

- **LilyPad Arduino**

An ATmega168 or ATmega132 running at 8 MHz with auto-reset, 6 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328P**

An ATmega328P running at 16 MHz with auto-reset. Equivalent to Arduino Duemilanove or Nano w/ ATmega328P; 6 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino NG or older w/ ATmega168**

An ATmega168 running at 16 MHz without auto-reset. Compilation and upload is equivalent to Arduino Diecimila or Duemilanove w/ ATmega168, but the bootloader burned has a slower timeout (and blinks the pin 13 LED three times on reset); 6 Analog In, 14 Digital I/O and 6 PWM.

- **Arduino Robot Control**

An ATmega328P running at 16 MHz with auto-reset.

- **Arduino Robot Motor**

An ATmega328P running at 16 MHz with auto-reset.

- **Arduino Gemma**

An ATtiny85 running at 8 MHz with auto-reset, 1 Analog In, 3 Digital I/O and 2 PWM.

INTRODUCTION TO EMBEDDED C

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.

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 languages:

- 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:

- 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). It is easier to write good code in C & convert it to an efficient assembly code 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.

CHAPTER 7

CONCLUSION

We presented a braille display based on electromagnetism principle. The main objective of research on the proposed display was to develop an actuator for implementing portable braille device for visually impaired people. The rotating structure gives the braille pin by the activating force and fixes it, so that the user can feel the same environment as the conventional braille system, and at the same time satisfies the low power consumption for the portable device.

REFERENCES

[1] G.DeviPriya, N.Indumathi, N.Kalaimagal, A.Suriya, J.T.Vasuki ,“Hardware Based Braille Pad on Mobile Phone” International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Special Issue 2, March 2015, ISSN(Online): 2320-9801,ISSN (Print): 2320- 9798,Dept. of Electronics and Communication, Narasu’sSarathy Institute of Technology, Salem, India.

[2] RumanSarkar, Smita Das, DwijenRudrapal ,“A low cost Micro electromechanical Braille for blind people to communicate with blind or deaf blind people through SMS subsystem” 2013 3rd IEEE International Advance Computing Conference (IACC), 978-1-4673-4529-3/12/$31.00_c 2012 IEEE, Computer Sc. & Engineering Department NIT Agartala, Tripura, India.

[3] Ms.Varsha Gaikwad, Prof. R. M. Khaire ,“Hardware Based Braille Note Taker” International Journal of Science, Engineering and Technology Research (IJSETR), Volume 4, Issue 11, November 2015, ISSN: 2278 – 7798.

[4] Zakaria Al-Qudah, Iyad Abu Doush, Faisal Alkhateeb, Esalm Al Maghayreh, Osama Al-Khaleel ,“Reading Braille on Mobile Phones: A Fast Method with Low Battery Power Consumption” 2011 International Conference on User Science and Engineering (i-USEr), 978-1-4577-1655- 3/11/$26.00 © 2011 IEEE, Computer Engineering Department Jordan University of Science and Technology, Irbid, Jordan.

[5]Jussi Rantala, Roope Raisamo, Jani Lylykangas, Veikko Surakka, Jukka Raisamo, Katri Salminen,Toni Pakkanen, and Arto Hippula ,“Methods for Presenting Braille Characters on a Mobile Device with a Touchscreen and Tactile Feedback” IEEE Transactions on Haptics, Vol. 2, No. 1, January-March 2009, 1939-1412/09/$25.00 _ 2009 IEEE.

[6]TanayChoudhary, SaurabhKulkarni, PradyumnaReddy ,“A Braille-Based Mobile Communication and Translation Glove for Deaf-blind People”

International Conference on Pervasive Computing (ICPC), 978-1-4799-6272-3/15/$31.00(c)2015 IEEE.Birla Institute of Technology & Science, Pilani - K.K. Birla Goa Campus Goa, India – 403726.

[7] Awang Damit, D. S., Che Ani, A. I., Muhamad, A. I., Abbas, M. H., & Ali, F. Z. (2014). Dual braille code translator: Basic education tool for visually impaired children. 2014 International Conference on Computer, Communications, and Control Technology (I4CT).

[8] Choudhary, T., Kulkarni, S., & Reddy, P. (2015). A Braille-based mobile communication and translation glove for deaf-blind people. 2015 International Conference on Pervasive Computing (ICPC).

[9] Shubhom, V. T., Keerthan, S., Swathi, S., Abhiram, G., & Shashidhar, R. (2017). BRAPTER: Compact braille transput communicator. 2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia).