A Mini Project Report On

**SMART GLOVES FOR DUMB AND DEAF**

*Submitted to CMREC (UGC Autonomous), Affiliated to JNTUH In Partial Fulfillment of the requirements for the Award of Degree of*

# BACHELOR OF TECHNOLOGY IN

**COMPUTER SCIENCE AND ENGINEERING (AI&ML)**

Submitted By

**K. PRIYANKA (218R1A66G4)**

**M. KAVYA (218R1A66G7)**

**M. PRAVEEN KUMAR (228R5A6616)**

**L. DHRUSHYATHI (228R5A6620)**

*Under the Esteemed guidance of*

## Mr. G. Venkateswarlu



**Assistant Professor, Department of CSE (AI & ML)**

**Department of Computer Science & Engineering (AI&ML)**

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

**2024-2025**

**CMR ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

*(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad) Kandlakoya, Medchal Road, Hyderabad-501 401*

**Department of Computer Science & Engineering (AI & ML)**



**CERTIFICATE**

This is to certify that the project entitled **“Smart gloves for dumb and deaf”** is a bonafide workcarried out by

**K. PRIYANKA (218R1A66G4)**

**M. KAVYA (218R1A66G7)**

**M. PRAVEEN KUMAR (228R5A6616)**

**L. DHRUSHYATHI (228R5A6620)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (AI&ML)** from **CMR**

**Engineering College**, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any otherdegree or diploma.

|  |  |  |
| --- | --- | --- |
| **Internal Guide** | **Project Coordinator** | **Head of the Department** |
| **Mr. G. Venkateswarlu** | **Mr. G. Venkateswarlu** | **Dr. M. Kumara Swamy** |
| Assistant Professor | Assistant Professor | Professor & HOD |
| CSE (AI&ML) | CSE (AI&ML) | CSE (AI&ML) |

**DECLARATION**

This is to certify that the work reported in the present Mini project entitled **“SMART GLOVES FOR DUMB AND DEAF”** is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI&ML), CMR Engineering College. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Mini project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

**K. PRIYANKA (218R1A66G4)**

**M. KAVYA (218R1A66G7)**

**M. PRAVEEN KUMAR (228R5A6616)** **L.DHRUSHYATHI (228R5A6620)**

**ACKNOWLEDGMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Kumara Swamy**, HOD,

**Department of CSE (AI & ML), CMR Engineering College** for their constant support.

We are extremely thankful to **Mr. G. Venkateswarlu,** Assistant Professor, Internal Guide, Departmentof CSE (AI & ML), for his constant guidance, encouragement and moral support throughout theproject.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mr. G. Venkateswarlu,** Mini Project Coordinator for his constant support incarrying out the project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

**K. PRIYANKA (218R1A66G4)**

**M. KAVYA (218R1A66G7)**

**M. PRAVEEN KUMAR (228R5A6616)**

**L. DHRUSHYATHI (228R5A6620)**

# INDEX

**TOPICS**

### Certificates………………………………………………………………………………………

* **Acknowledgement…………………………………………………………………………........**

**CHAPTER 1: INTRODUCTION**

* 1. **Introduction of the project …………………………………………………………………………………**
  2. **Project overview……………………………………………………………………………………………...**
  3. **Thesis…………………………………………………………………………………………………………**

**CHAPTER 2: EMBEDDED SYSTEMS**

* 1. **Introduction to embedded systems…………………………………………………………………………**
  2. **Need of embedded systems…………………………………………………………………………………...**
  3. **Explanation of embedded systems…………………………………………………………………………...**
  4. **Applications of embedded systems…………………………………………………………………………**

**CHAPTER 3: HARDWARE DESCRIPTION**

* 1. **Introduction with block diagram……………………………………………………………………………**
  2. **Microcontroller……………………………………………………………………………………………….**
  3. **Regulated power supply……………………………………………………………………………………...**
  4. **LED……………………………………………………………………………………………………………**
  5. **FLEX sensor……….………………………………………………………………………………………**
  6. **RF transmitter and RF receiver …….………………………………………………….............................**
  7. **VOICE MODULE…………...………………………………………………………………………**
  8. **Relay ...……………………………………………………………………………………………**

**CHAPTER 4: SOFTWARE DESCRIPTION**

* 1. **Express PCB…………………………………………………………………………………………………**
  2. **PIC C Compiler……………………………………………………………………………………………….**
  3. **Proteus software………………………………………………………………………………………………**

### Procedural steps for compilation, simulation and dumping……………………………………..

**CHAPTER 5: PROJECT DESCRIPTION**

**CHAPTER 6: ADVANTAGES, DISADVANTAGES AND APPLICATIONS**

**CHAPTER 7: RESULTS, CONCLUSION, FUTURE PROSPECTS**

**REFERENCES**

# CHAPTER 1: INTRODUCTION

### Introduction:

The main aim of this project is to design and construct a system through which different voices can be announced using flex sensors.

Arduino based speaking system for deaf and dumb is designed to give the signs, which are preloaded in the device. It is a micro controller based device, which gives the alert sounds just by bending the flex sensor, which are given some predefined messages like asking for water, food, washroom etc. Here the person can just folding the flex sensors which indicates the sign of water (example) then the device sounds the same with some output volume.

This flex sensor is a unique component that changes resistance in proportion to the degree it is bent. The sensor when lying flat has a nominal resistance. As the flex sensor is bent the resistance increases in proportion. This device is very helpful for paralysis and physically challenged persons.

The whole system is controlled by Arduino UNO microcontroller. Three flex sensors and Voice module are connected to the ARDUINO microcontroller. User need to wear the glove to his/her hand. We are placing flex sensors fingers of the glove. When the user folding the flex sensors this particular resistance read by Arduino and announce the appropriate voices through speaker. The Microcontroller is loaded with an intelligent program written using embedded ‘C’ language.

### Project Overview:

An embedded system is a combination of software and hardware to perform a dedicated task*.* Some of the main devices used in embedded products are Microprocessors and Microcontrollers.

Microprocessors are commonly referred to as general purpose processors as they simply accept the inputs, process it and give the output. In contrast, a microcontroller not only accepts the data as inputs but also manipulates it, interfaces the data with various devices, controls the data and thus finally gives the result.

The project **Smart Gloves for Dumb and Deaf** using Arduino Microcontroller is an exclusive project that can announce the voices of user basic needs by folding flex sensors.

### Thesis Overview:

The thesis explains the implementation of “**Smart Gloves for Dumb and Deaf**” using ARDUINO. The organization of the thesis is explained here with:

**Chapter 1** Presents introduction to the overall thesis and the overview of the project. In the project overview a brief introduction of “**Smart Gloves for Dumb and Deaf** “and its applications are discussed

**Chapter 2** Presents the topic embedded systems. It explains the about what is embedded systems, need for embedded systems, explanation of it along with its applications.

**Chapter 3** Presents the hardware description. It deals with the block diagram of the project and explains the purpose of each block. In the same chapter the explanation of microcontroller, power supplies, FLEX sensor, VOICE MODULE are considered.

**Chapter 4** Presents the software description. It explains the implementation of the project using ARDUINO IDE Compiler software.

**Chapter 5 Presents** the project description along with FLEX sensor, VOICE MODULE interfacing to microcontroller.

**Chapter 6** Presents the advantages, disadvantages and applications of the project.

**Chapter 7** Presents the results, conclusion and future scope of the project.

# CHAPTER 2: EMBEDDED SYSTEMS

### Embedded Systems:

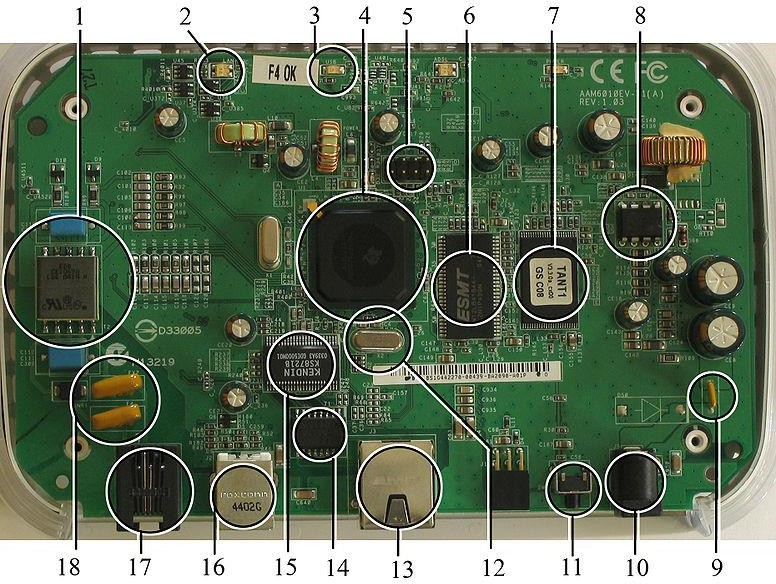
An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a

personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites. (Each radar probably includes one or more embedded systems of its own.)

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.



### Fig 2.1:A modern example of embedded system

Labeled parts include microprocessor (4), RAM (6), flash memory (7).Embedded systems programming is not like normal PC programming. In many ways, programming for an embedded system is like programming PC 15 years ago. The hardware for the system is usually chosen to make the device as cheap as possible. Spending an extra dollar a unit in order to make things easier to program can cost millions. Hiring a programmer for an extra month is cheap in comparison. This means the programmer must make do with slow processors and low memory, while at the same time battling a need for efficiency not seen in most PC applications. Below is a list of issues specific to the embedded field.

### History:

In the earliest years of computers in the 1930–40s, computers were sometimes

dedicated to a single task, but were far too large and expensive for most kinds of tasks performed by embedded computers of today. Over time however, the concept of [programmable controllers](http://en.wikipedia.org/wiki/Programmable_controllers) evolved from traditional [electromechanical](http://en.wikipedia.org/wiki/Electromechanical) sequencers, via solid state devices, to the use of computer technology.

One of the first recognizably modern embedded systems was the [Apollo Guidance](http://en.wikipedia.org/wiki/Apollo_Guidance_Computer) [Computer](http://en.wikipedia.org/wiki/Apollo_Guidance_Computer), developed by [Charles Stark Draper](http://en.wikipedia.org/wiki/Charles_Stark_Draper) at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An

early mass-produced embedded system was the Autonetics D-17 guidance computer for

the [Minuteman missile,](http://en.wikipedia.org/wiki/Minuteman_(missile)) released in 1961. It was built from [transistor](http://en.wikipedia.org/wiki/Transistor) [logic](http://en.wikipedia.org/wiki/Digital_circuit) and had a [hard disk](http://en.wikipedia.org/wiki/Hard_disk) for main memory. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits.

### Tools:

Embedded development makes up a small fraction of total programming. There's also a large number of embedded architectures, unlike the PC world where 1 instruction set rules, and the UNIX world where there's only 3 or 4 major ones. This means that the tools are more expensive. It also means that they're lowering featured, and less developed. On a major embedded project, at some point you will almost always find a compiler bug of some sort.

Debugging tools are another issue. Since you can't always run general programs on your embedded processor, you can't always run a debugger on it. This makes fixing your program difficult. Special hardware such as JTAG ports can overcome this issue in part. However, if you stop on a breakpoint when your system is controlling real world hardware (such as a motor), permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

### Resources:

To save costs, embedded systems frequently have the cheapest processors that can do the job. This means your programs need to be written as efficiently as possible. When dealing with large data sets, issues like memory cache misses that never matter in PC programming can hurt you. Luckily, this won't happen too often- use reasonably efficient algorithms to start, and optimize only when necessary. Of course, normal profilers won't work well, due to the same reason debuggers don't work well.

Memory is also an issue. For the same cost savings reasons, embedded systems usually have the least memory they can get away with. That means their algorithms must be memory efficient (unlike in PC programs, you will frequently sacrifice processor time for memory, rather than the reverse). It also means you can't afford to leak memory. Embedded applications generally use deterministic memory techniques and avoid the default "new" and "malloc" functions, so that leaks can be found and eliminated more easily. Other resources programmers expect may not even exist.

### Real Time Issues:

Embedded systems frequently control hardware, and must be able to respond to them in real time. Failure to do so could cause inaccuracy in measurements, or even damage hardware such as motors. This is made even more difficult by the lack of resources available. Almost all embedded systems need to be able to prioritize some tasks over others, and to be able to put off/skip low priority tasks such as UI in favor of high priority tasks like hardware control.

### Need For Embedded Systems:

The uses of embedded systems are virtually limitless, because every day new products are introduced to the market that utilizes embedded computers in novel ways. In recent years, hardware such as microprocessors, microcontrollers, and FPGA chips have become much cheaper. So when implementing a new form of control, it's wiser to just buy the generic chip and write your own custom software for it. Producing a custom-made chip to handle a particular task or set of tasks costs far more time and money. Many embedded computers even come with extensive libraries, so that "writing your own software" becomes a very trivial task indeed. From an implementation viewpoint, there is a major difference between a computer and an embedded system. Embedded systems are often required to provide Real-Time response. The main elements that make embedded systems unique are its reliability and ease in debugging.

### Debugging:

Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticate they can be roughly grouped into the following areas:

* + - * Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
      * External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multi core systems.
      * An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
      * An in-circuit emulator replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
      * Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as assembly code or source-code.

### Reliability:

Embedded systems often reside in machines that are expected to run continuously for years without errors and in some cases recover by them 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:

* + - * 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.
      * The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backup s is selected by an operator. Examples include aircraft navigation, reactor control

systems, safety-critical chemical factory controls, train signals, engines on single-engine aircraft.

* + - * 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.

### Explanation of Embedded Systems:

* + 1. **Software Architecture:**

There are several different types of software architecture in common use.

* + - * Simple Control Loop:

In this design, the software simply has a loop. The loop calls subroutines, each of which manages a part of the hardware or software.

* + - * Interrupt Controlled System:

Some embedded systems are predominantly interrupt controlled. This means that tasks performed by the system are triggered by different kinds of events. An interrupt could be generated

for example by a timer in a predefined frequency, or by a serial port controller receiving a byte. These kinds of systems are used if event handlers need low latency and the event handlers are short and simple.

Usually these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays. Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop.

This method brings the system close to a multitasking kernel with discrete processes.

* + - * Cooperative Multitasking:

A non-preemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for no operation), etc.The advantages and disadvantages are very similar to the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue-interpreter.

* + - * Primitive Multitasking:

In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

### Stand Alone Embedded System:

These systems takes the input in the form of electrical signals from transducers or commands from human beings such as pressing of a button etc.., process them and produces desired output. This entire process of taking input, processing it and giving output is done in standalone mode. Such embedded systems comes under stand alone embedded systems

Eg: microwave oven, air conditioner etc..

### Real-time embedded systems:

Embedded systems which are used to perform a specific task or operation in a specific time period those systems are called as real-time embedded systems. There are two types of real-time embedded systems.

* + - * Hard Real-time embedded systems:

These embedded systems follow an absolute dead line time period i.e.., if the tasking is not done in a particular time period then there is a cause of damage to the entire equipment.

Eg: consider a system in which we have to open a valve within 30 milliseconds. If this valve is not opened in 30 ms this may cause damage to the entire equipment. So in such cases we use embedded systems for doing automatic operations.

* + - * Soft Real Time embedded systems:

Eg: Consider a TV remote control system, if the remote control takes a few milliseconds delay it will not cause damage either to the TV or to the remote control. These systems which will not cause damage when they are not operated at considerable time period those systems comes under soft real-time embedded systems.

### Network communication embedded systems:

A wide range network interfacing communication is provided by using embedded

systems.

Eg:

* + - * Consider a web camera that is connected to the computer with internet can be used to spread communication like sending pictures, images, videos etc.., to another computer with internet connection throughout anywhere in the world.

Whenever a person comes near the door, it captures the image of a person and sends to the desktop of your computer which is connected to internet. This gives an alerting message with image on to the desktop of your computer, and then you can open the door lock just by clicking the mouse. Fig: 2.2 show the network communications in embedded systems.



### Fig 2.2: Network communication embedded systems

* + 1. **Different types of processing units:**

The central processing unit (c.p.u) can be any one of the following microprocessor, microcontroller, digital signal processing.

* + - * Among these Microcontroller is of low cost processor and one of the main advantage of microcontrollers is, the components such as memory, serial communication interfaces, analog to digital converters etc.., all these are built on a single chip. The numbers of external components that are connected to it are very less according to the application.
      * Microprocessors are more powerful than microcontrollers. They are used in major applications with a number of tasking requirements. But the microprocessor requires many external components like memory, serial communication, hard disk, input output ports etc.., so the power consumption is also very high when compared to microcontrollers.
      * Digital signal processing is used mainly for the applications that particularly involved with processing of signals
  1. **APPLICATIONS OF EMBEDDED SYSTEMS:**

### Consumer applications:

At home we use a number of embedded systems which include microwave oven, remote control, vcd players, dvd players, camera etc….



### Fig2.3: Automatic coffee makes equipment

* + 1. **Office automation:**

We use systems like fax machine, modem, printer etc…



### Fig2.4: Fax machine Fig2.5: Printing machine

### 2.4.5 Computer networking:

Embedded systems are used as bridges routers etc..

**Fig2.7: Computer networking**

### 2.4.6 Tele communications:

Cell phones, web cameras etc.

**Fig2.8: Cell Phone Fig2.9: Web camera**

# CHAPTER 3: HARDWARE DESCRIPTION

### Introduction:

In this chapter the block diagram of the project and design aspect of independent modules are considered. Block diagram is shown in fig: 3.1:

### FIG 3.1(i): Block diagram of smart gloves for dumb and deaf

**The main blocks of this project are**:

* + - Adapter power supply.
    - Arduino UNO.
    - Three Flex sensors.
    - Voice circuit.

### Micro controller:



**Fig: 3.2 Arduino uno atmega328p Microcontroller**

⯈ The Arduino Uno is a microcontroller board which has ATmega328 from the AVR family.

There are 14 digital input/output pins, 6 Analog pins and 16MHz ceramic resonator.

⯈ USB connection, power jack and also a reset button is used. Its software is supported by a number of libraries that makes the programming easier.

**ATMEGA328:**

**Features**

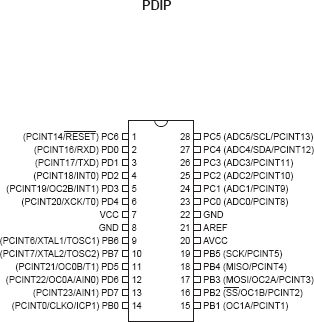
* High Performance, Low Power AVR® 8-Bit Microcontroller
* Advanced RISC Architecture
* 131 Powerful Instructions – Most Single Clock Cycle Execution
* 32 x 8 General Purpose Working Registers
* Fully Static Operation
* Up to 20 MIPS Throughput at 20 MHz
* On-chip 2-cycle Multiplier
* High Endurance Non-volatile Memory Segments
* 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
* 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
* 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
* Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
* Data retention: 20 years at 85°C/100 years at 25°C(1)
* Optional Boot Code Section with Independent Lock Bits In-System Programming by On-chip Boot Program

True Read-While-Write Operation

* Programming Lock for Software Security
* **Peripheral Features**
* Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
* One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
* Real Time Counter with Separate Oscillator
* Six PWM Channels
* 8-channel 10-bit ADC in TQFP and QFN/MLF package

Temperature Measurement

* 6-channel 10-bit ADC in PDIP Package Temperature Measurement
* Programmable Serial USART
* Master/Slave SPI Serial Interface
* Byte-oriented 2-wire Serial Interface (Philips I2C compatible)
* Programmable Watchdog Timer with Separate On-chip Oscillator
* On-chip Analog Comparator
* Interrupt and Wake-up on Pin Change
* **Special Microcontroller Features**
* Power-on Reset and Programmable Brown-out Detection
* Internal Calibrated Oscillator
* External and Internal Interrupt Sources
* Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
* I/O and Packages
* 23 Programmable I/O Lines
* 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
* Operating Voltage:
* 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
* Temperature Range:
* -40°C to 85°C
* Speed Grade:
* 0 - 20 MHz @ 1.8 - 5.5V
* Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
* Active Mode: 0.2 mA
* Power-down Mode: 0.1 μA
* Power-save Mode: 0.75 μA (Including 32 kHz RTC)



* 1. **Pin Descriptions**
     1. **VCC** Digital supply voltage.
     2. **GND** Ground.
     3. **Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2**

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each it). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri- stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set. The various special features of Port B are elaborated in ”Alternate Functions of Port B” on page 82 and ”System Clock and Clock Options” on page 26.

* + 1. **Port C (PC5:0)**

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each it). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri- stated when a reset condition becomes active, even if the clock is not running.

* + 1. **PC6/RESET**

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is un programmed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running.

The minimum pulse length is given in Table 28-3 on page 318. Shorter pulses are not guaranteed to generate a Reset. The various special features of Port C are elaborated in ”Alternate Functions of Port C” on page 85.

* + 1. **Port D (PD7:0)**

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri- stated when a reset condition becomes active, even if the clock is not running. The various special features of Port D are elaborated in ”Alternate Functions of Port D” on page

88.

* + 1. **AVCC**

AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

* + 1. **AREF**

AREF is the analog reference pin for the A/D Converter.

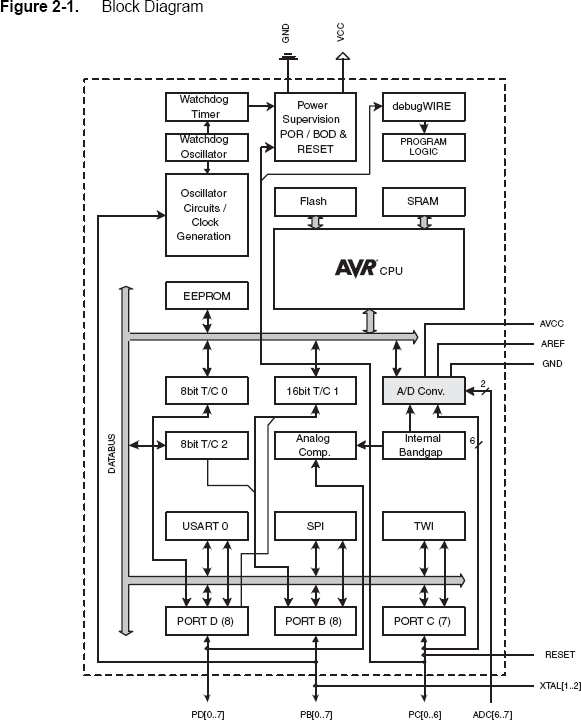
* + 1. **ADC7:6 (TQFP and QFN/MLF Package Only)**

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

**Overview**

The ATmega48PA/88PA/168PA/328P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48PA/88PA/168PA/328P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

* 1. **Block Diagram**



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture 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: 4K/8K 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

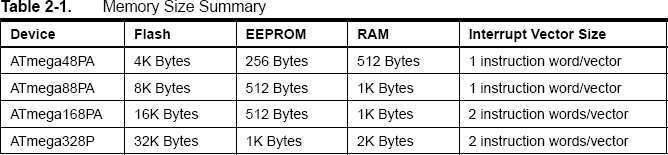
Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, 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 will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self- Programmable Flash on a monolithic 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.

The ATmega48PA/88PA/168PA/328P AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

* 1. **Comparison Between ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P**



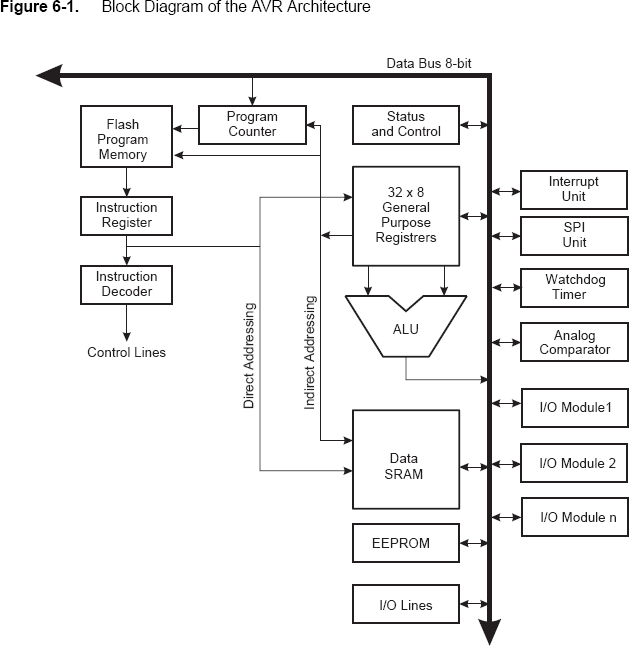
The ATmega48PA, ATmega88PA, ATmega168PA and ATmega328P differ only in memory sizes, boot loader support, and interrupt vector sizes. Table 2-1 summarizes the different memory and interrupt vector sizes for the three devices.

ATmega88PA, ATmega168PA and ATmega328P support a real Read-While-Write Self-Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In ATmega48PA, there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash.

**AVR CPU Core**

* 1. **Overview**

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In- System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle. Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position.

The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers,

SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the ATmega48PA/88PA/168PA/328P has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

* 1. **ALU – Arithmetic Logic Unit**

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful

multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

* 1. **Status Register**

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code. The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

**AVR Memories**

* 1. **Overview**

This section describes the different memories in the ATmega48PA/88PA/168PA/328P. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega48PA/88PA/168PA/328P features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

* 1. **In-System Reprogrammable Flash Program Memory**

The ATmega48PA/88PA/168PA/328P contains 4/8/16/32K bytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 2/4/8/16K x 16. For software security, the Flash Program memory space is divided into two sections, Boot Loader Section and Application Program Section in ATmega88PA and ATmega168PA. See SELFPRGEN description in section ”SPMCSR – Store Program Memory Control and Status Register” on page 292 for more details.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction description).

**SRAM Data Memory**

The ATmega48PA/88PA/168PA/328P is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used. The lower 768/1280/1280/2303 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512/1024/1024/2048 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space. The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

**EEPROM Data Memory**

The ATmega48PA/88PA/168PA/328P contains 256/512/512/1K bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

* + 1. **EEPROM Read/Write Access**

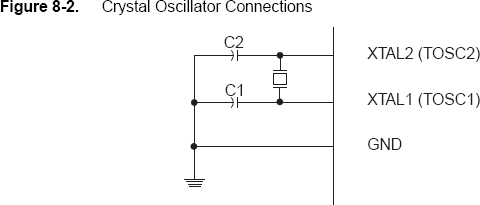
The EEPROM Access Registers are accessible in the I/O space.

lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this. When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

**Low Power Crystal Oscillator**

Pins XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, Either a quartz crystal or a ceramic resonator may be used. This Crystal Oscillator is a low power oscillator, with reduced voltage swing on the XTAL2 output.

It gives the lowest power consumption, but is not capable of driving other clock inputs, and may be more susceptible to noise in noisy environments.



**Watchdog Timer Features**

* Clocked from separate On-chip Oscillator
* 3 Operating modes
* Interrupt
* System Reset
* Interrupt and System Reset
* Selectable Time-out period from 16ms to 8s
* Possible Hardware fuse Watchdog always on (WDTON) for fail-safe mode

**Overview**

ATmega48PA/88PA/168PA/328P has an Enhanced Watchdog Timer (WDT). The WDT is a timer counting cycles of a separate on-chip 128 kHz oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

The Watchdog always on (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed the System Reset mode bit (WDE) and Interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing WDE and changing time-out configuration is as follows:

1. In the same operation, write a logic one to the Watchdog change enable bit (WDCE) and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and Watchdog prescaler bits (WDP) as desired, but with the WDCE bit cleared. This must be done in one operation.

**8-bit Timer/Counter0 with PWM Features**

* + Two Independent Output Compare Units
  + Double Buffered Output Compare Registers
  + Clear Timer on Compare Match (Auto Reload)
  + Glitch Free, Phase Correct Pulse Width Modulator (PWM)
  + Three Independent Interrupt Sources (TOV0, OCF0A, and OCF0B)
  1. **ADAPTER POWER SUPPLY:**

## Adapter

The AC adapter, AC/DC adapter or AC/DC converter is a type of external [power supply,](https://en.wikipedia.org/wiki/Power_supply) often enclosed in a case similar to an [AC plug.](https://en.wikipedia.org/wiki/AC_power_plugs_and_sockets) Other names include [plug pack,](https://en.wikipedia.org/wiki/Plug_pack) plug-in adapter, adapter block, domestic mains adapter, line power adapter, wall wart, or power adapter. AC adapters are used with electrical devices that require power but do not contain internal components to derive the required voltage and power from [mains power](https://en.wikipedia.org/wiki/Mains_electricity). The internal circuitry of an external power supply is very similar to the design that would be used for a built-in or internal supply.



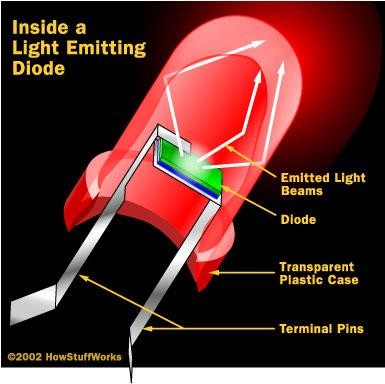
External power supplies are used both with equipment with no other source of power and with [battery](https://en.wikipedia.org/wiki/Battery_%28electricity%29)-powered equipment, where the supply, when plugged in, can sometimes charge the battery in addition to powering the equipment.

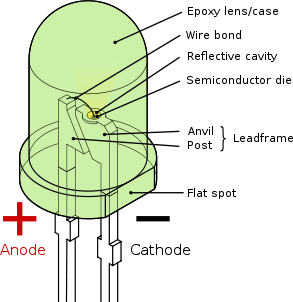
Use of an external power supply allows portability of battery-powered equipment without the added bulk of internal power components and makes it unnecessary to produce equipment for use only with a specified power source



* must be a DC adapter (i.e. it has to put out DC, not AC);
* should be between 9V and 12V DC (see note below);
* must be rated for a minimum of 250mA current output, although you will likely want something more like 500mA or 1A output, as it gives you the current necessary to power a servo or twenty LEDs if you want to.
* must have a 2.1mm power plug on the Arduino end.

**3.4. LED:**

A light-emitting diode (LED) is a semiconductor light source. LED’s are used as indicator lamps in many devices, and are increasingly used for lighting. Introduced as a practical electronic component in 1962, early LED’s emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness. The internal structure and parts of a led are shown in figures 3.4.1 and 3.4.2 respectively.



### Fig 3.4.1: Inside a LED Fig 3.4.2: Parts of a LED

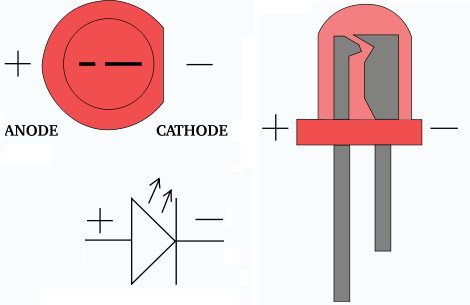
**Working:**

The structure of the LED light is completely different than that of the light bulb. Amazingly, the LED has a simple and strong structure. The light-emitting semiconductor material is what determines the LED's color. The LED is based on the semiconductor diode.

When a diode is forward biased (switched on), electrons are able to recombine with holes within the device, releasing energy in the form of photons. This effect is called electroluminescence and the color of the light (corresponding to the energy of the photon) is determined by the energy gap of the semiconductor. An LED is usually small in area (less than 1 mm2), and integrated optical components are used to shape its radiation pattern and assist in reflection. LED’s present many advantages over incandescent light sources including lower energy consumption, longer lifetime, improved robustness, smaller size, faster switching, and greater

durability and reliability. However, they are relatively expensive and require more precise current and

heat management than traditional light sources. Current LED products for general lighting are more expensive to buy than fluorescent lamp sources of comparable output. They also enjoy use in applications as diverse as replacements for traditional light sources in automotive lighting (particularly indicators) and in traffic signals. The compact size of LED’s has allowed new text and video displays and sensors to be developed, while their high switching rates are useful in advanced communications technology. The electrical symbol and polarities of led are shown in fig: 3.4.3.

LED lights have a variety of advantages over other light sources:

* High-levels of brightness and intensity
* High-efficiency
* Low-voltage and current requirements
* Low radiated heat
* High reliability (resistant to shock and vibration)
* No UV Rays
* Long source life
* Can be easily controlled and programmed Applications of LED fall into three major categories:
* Visual signal application where the light goes more or less directly from the LED to the human eye, to convey a message or meaning.
* Illumination where LED light is reflected from object to give visual response of these objects.

**3.5 FLEX SENSOR:**

**Introduction:**

The Flex Sensor patented technology is based on resistive carbon elements. As a variable printed resistor, the Flex Sensor achieves great form-factor on a thin flexible substrate. When the substrate is bent, the sensor produces a resistance output correlated to the bend radius—the smaller the radius, the higher the resistance value.

In 1982 Thomas G. Zimmerman filed a patent (US Patent 4542291) on an optical flex sensor mounted in a glove to measure finger bending. Zimmerman worked with Jaron Lanier to incorporate ultrasonic and magnetic hand position tracking technology to create the Power Glove and Data Glove, respectively (US Patent 4988981, filed 1989). The optical flex sensor used in the Data Glove was invented by Young **L. Harvill** (US Patent 5097252, filed 1989) who scratched the fiber near the finger joint to make it locally sensitive to bending.

A wired glove is a glove-like input device for human-computer interaction, often in virtual reality environments. Various sensor technologies are used to capture physical data such as bending of fingers. Often a motion tracker, such as a magnetic tracking device or inertial tracking device, is attached to capture the global position/rotation data of the glove. These movements are then interpreted by the software that accompanies the glove, so any one movement can mean any number of things.

Expensive high-end wired gloves can also provide haptic feedback, which is a simulation of the sense of touch. This allows a wired glove to also be used as an output device. Traditionally, wired gloves have only been available at a huge cost, with the finger bend sensors and the tracking device having to be bought separately.

Wired gloves are often called "datagloves" or "cybergloves". The latter term is a trademark of Immersion Corporation (which acquired Virtual Technologies, Inc. and its patent portfolio in September 2000).

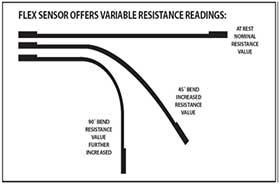
An alternative to wired gloves is to use a camera and computer vision to track the 3D pose and trajectory of the hand, at the cost of tactile feedback.

**Description:**

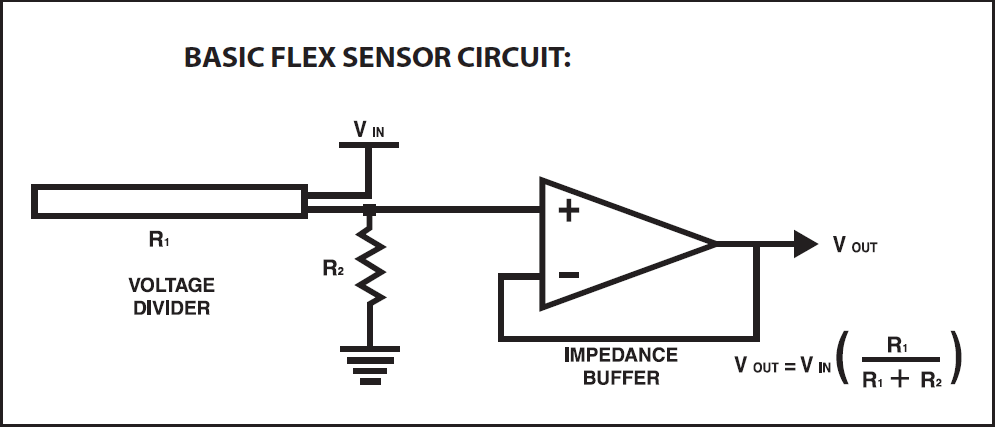
### Working of flex sensor:

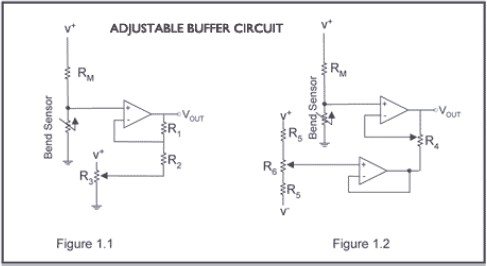
Flex sensor are sensors that change in resistance depending on the amount of bend on the sensor. They convert the change in bend to electrical resistance the more bend the more the resistance value. They are usually in the form of a thin strip from 1”-5” long that vary in resistance. They can be made uni-directional and bi-directional.

|  |  |  |  |
| --- | --- | --- | --- |
| Sizes - | 1K | to | 20K |
|  | 50K | to | 50ohm |
|  | 50K | to | 200k |

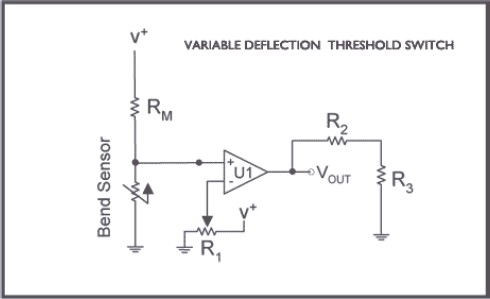


### Basic circuit operation of the flex sensor:

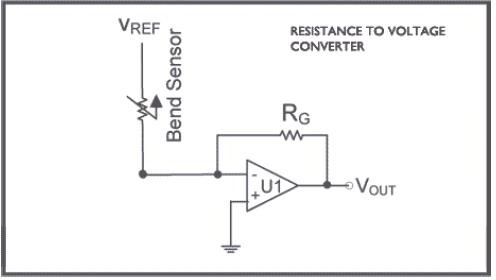


"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces errer due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324." "You can also test your flex sensor using the simplest circuit, and skip the op amp." **"Adjustable Buffer** - a potentiometer can be added to the circuit to adjust the sensitivity range."

**`"Variable Deflection Threshold Switch** - an op amp is used and outputs either high or low depending on the voltage of the inverting input. In this way you can use the flex sensor as a switch without going through a microcontroller."



**"Resistance to Voltage Converter** - use the sensor as the input of a resistance to voltage converter using a dual sided supply op-amp. A negative reference voltage will give a positive output. Should be used in situations when you want output at a low degree of bending"



**ATTRIBUTES:**

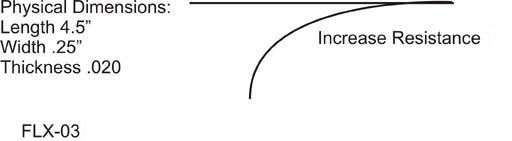
Custom designed to match customer specs High level of reliability, consistency, repeatability Harsh temperature resistance Variety of flexible or stationary surfaces for mounting Infinite number of resistance possibilities and bend ratios Please call our full Design Engineering team or Sales Engineers for any questions or ideas at One directional Flex Sensor is a unique component that changes resistance when bent or flexed (Patent 5086785). A UN flexed sensor has a nominal resistance of approximately 10,000 ohms (10 K). As the flex sensor is bent in one direction the resistance gradually increases.

The flex sensor operating temperature is -45F to 125F.

Bi-Directional Flex Sensor

Connection - Pin connector .100" spacing

The sensor measures 1/4 inch wide, 4 1/2 inches long and only .019 inches thick!



**FEATUTRES:**

* Size: approx 0.28" wide and 1"/3"/5" long
* Resistance Range: 1.5-40K ohms depending on sensor. Flex point claims a 0-250 resistance range.
* Lifetime: Greater than 1 million life cycles
* Temperature Range: -35 to +80 degrees Celsius
* Hysteresis: 7%
* Voltage: 5 to 12 V

**SPECIFICATION:**

|  |  |
| --- | --- |
| **parameter** | **value** |
| Operating Voltage | +5V DC regulated |

|  |  |
| --- | --- |
| Operating Current | 100mA |
| Output | Analog output |
| Flex Bending | Nominal,45 degree,90 degree |

**Mechanical Specifications:**

1. Life Cycle: >1 million
2. Height: 0.43mm (0.017")
3. Temperature Range: -35°C to +80°C

**Electrical Specifications:**

1. Flat Resistance: 10K Ohms
2. Resistance Tolerance: ±30%
3. Bend Resistance Range: 60K to 110K Ohms
4. Power Rating: 0.50 Watts continuous. 1 Watt Peak

**Advantages of the flex sensor:**

* + **Environmental impact.** Ethanol burns cleaner than gasoline and therefore is responsible for fewer toxic fumes, which is highly advantageous from an anti-pollution point of view. The fact that ethanol does not contribute significantly to greenhouse gasses, makes it a popular alternative among the environmentally conscious
  + **Burning facility.** Possibly the greatest advantage is that the flex fuel vehicle has been designed to burn whatever proportion of mixture is in its combustion chamber. Electronic sensors gauge the blend, while microprocessors adjust the fuel injection and timing.
  + **Alternative to oil.** Many flex fuel vehicles make use of ethanol, which originates from corn and sugar cane, a viable alternative to purchasing foreign oil.

**Uses:**

Spectra Symbol has used this technology in supplying Flex Sensors for the Nintendo Power Glove, the P5 gaming glove, and the below applications:

* Automotive controls
* Medical devices
* Industrial controls
* Computer peripherals
* Fitness products
* Musical instruments
* Measuring devices
* Virtual reality games
* Consumer products
* Physical therapy
* To test sensor you only need power the sensor by connect two wires +5V and GND. You can leave the output wire as it is.
* The flex sensor analog resistors. They work as analog voltage dividers.
* When the substrate is bend the sensor produces resistance output relative to bending radius nominal, 45 degree and 90 degree.
* Measure the output voltage through multi-meter between OUT and Ground pins or Use a microcontroller to measure the voltage output.

### Some applications for the Flex Sensor are:

* + Collision detection on mobile robots
  + VR Gloves and VR suits
  + Physics applications and experiments
  + gaming gloves
  + auto controls
  + fitness products
  + measuring devices
  + assistive technology
  + musical instruments and joysticks

# 3.6 VOICE MODULE APR 33A3:

## Features:

* Operating Voltage Range: 3V ~ 6.5V
* Single Chip, High Quality Audio/Voice Recording & Playback Solution
* No External ICs Required
* Powerful 16-Bits Digital Audio Processor.
* Nonvolatile Flash Memory Technology
* No Battery Backup Required
* External Reset pin.
* Powerful Power Management Unit
* Very Low Standby Current: 1uA
* Low Power-Down Current: 15uA
* Supports Power-Down Mode for Power Saving
* Built-in Audio-Recording Microphone Amplifier
* No External OPAMP or BJT Required
* Easy to PCB layout
* Configurable analog interface
* Differential-ended MIC pre-amp for Low Noise
* High Quality Line Receiver
* High Quality Analog to Digital and PWM module
* Resolution up to 16-bits
* Simple And Direct User Interface
* Averagely 1,2,4 or 8 voice messages record & playback

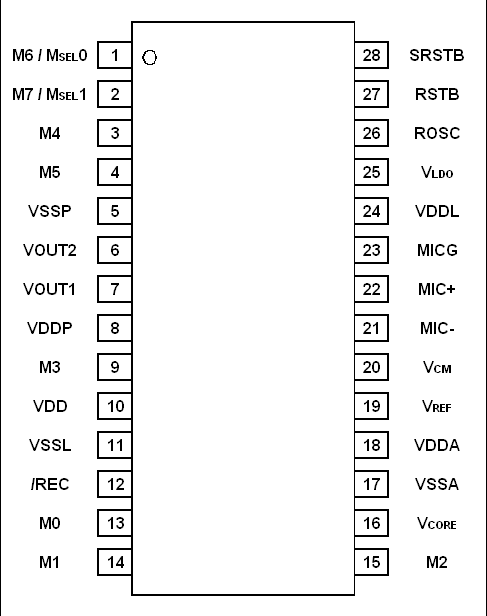
# DESCRIPTION

The aPR33A series are powerful audio processor along with high performance audio analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). The aPR33A series are a fully integrated solution offering high performance and unparalleled integration with analog input, digital processing and analog output functionality.

The aPR33A series incorporates all the Functionality required to perform demanding audio/voice applications. Functionality required to perform demanding audio/voice applications. High quality audio/voice systems with lower bill-of-material costs can be implemented with the aPR33A series because of its integrated analog data converters and full suite of quality- enhancing features.

The aPR33A series C2.0 is specially designed for simple key trigger, user can record and playback the message averagely for 1, 2, 4 or 8 voice message(s) by switch, It is suitable in simple interface or need to limit the length of single message, e.g. toys, leave messages system, answering machine etc. Meanwhile, this mode provides the power-management system. Users can let the chip enter power-down mode when unused. It can effectively reduce electric current consuming to 15uA and increase the using time in any projects powered by batteries.

# PIN CONFIGURATION



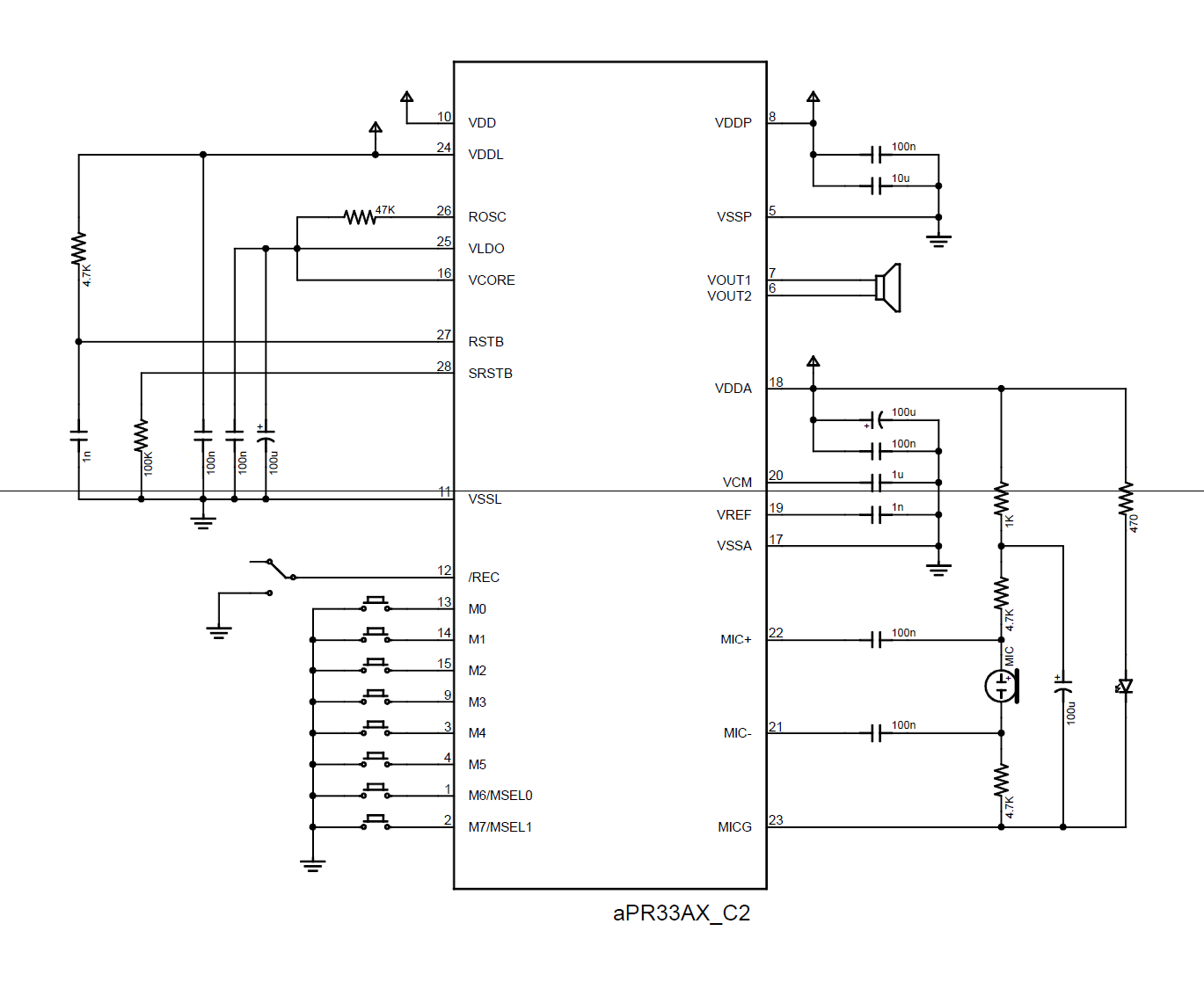
**SOP Package**

# PIN DESCRIPTION

|  |  |  |
| --- | --- | --- |
| Pin Names | TYPE | Description |
| VDDP VDD VDDA VDDL |  | Positive power supply. |
| VSSP VSSL VSSA |  | Power ground. |
| VLDO |  | Internal LDO output. |
| VCORE |  | Positive power supply for core. |
| VREF |  | Reference voltage. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| VCM |  | Common mode voltage. |
| Rosc | INPUT | Oscillator resistor input. |
| RSTB | INPUT | Reset. (Low active) |
| SRSTB | INPUT | System reset, pull-down a resistor to the VSSL. |
| MIC+ MIC- | INPUT | Microphone differential input. |
| MICG | OUTPUT | Microphone ground. |
| VOUT2 VOUT1 | OUTPUT | PWM output to drive speaker directly. |
| REC | INPUT | Record Mode. (Low active) |
| M0 | INPUT | Message-0. |
| M1 | INPUT | Message-1. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| M2 | INPUT | Message-2. |
| M3 | INPUT | Message-3 |
| M4 | INPUT | Message-4 |
| M5 | INPUT | Message-5 |
| M6 / MSEL0 | INPUT | Message-6, Message select 0. |
| M7 / MSEL1 | INPUT | Message-7, Message select 1. |

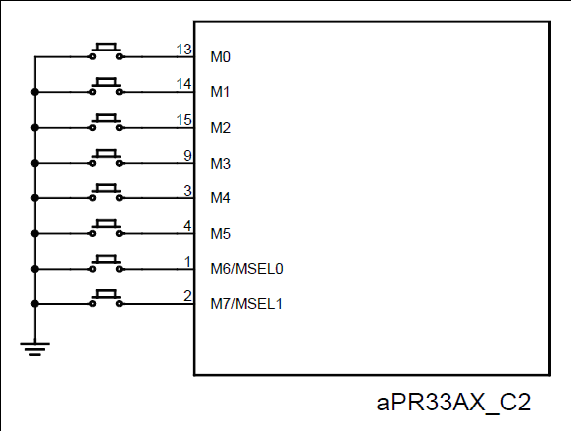


**MESSAGE MODE**

In fixed 1/ 2/ 4/ 8 message mode (C2.0), user can divide the memory averagely for 1, 2, 4 or 8 message(s). The message mode will be applied after chip reset by the MSEL0 and MSEL1 pin. Please note the message should be recorded and played in same message mode, we CAN NOT guarantee the message is complete after message mode changed. For example, user recorded 8 messages in the 8-message mode, those messages can be played in 8-message mode only. If user changed to 1, 2 or 4 message mode, system will discard those messages.

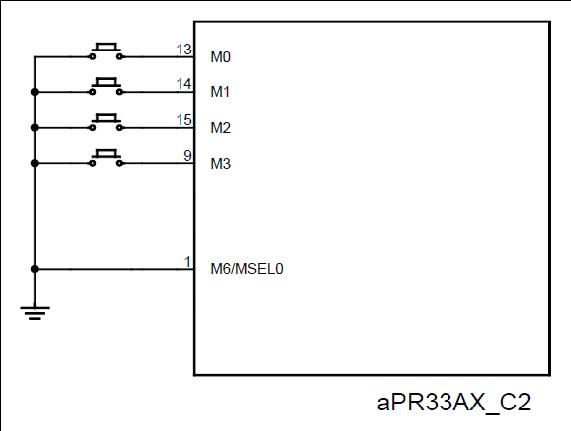
 8-Message Mode

The memory will be divided to 8 messages averagely when both MSEL0 and MSEL1 pin float after chip reset.

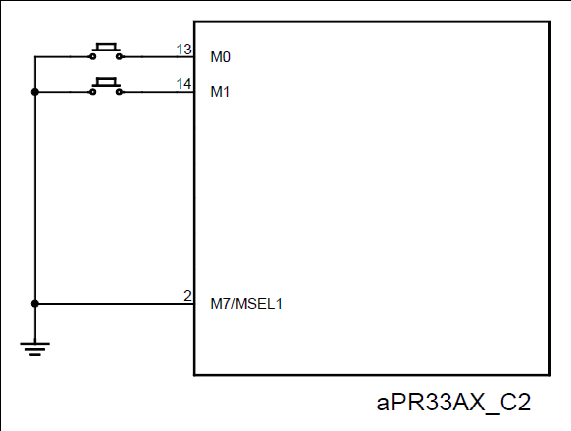


1. Message Mode

The memory will be divided to 4 messages averagely when MSEL0 pin connected to VSS and MSEL1 pin float after chip reset.

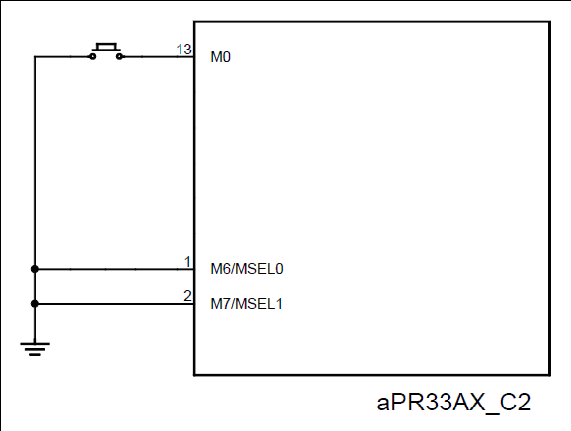


2-Message Mode

The memory will be divided to 2 messages averagely when MSEL1 pin connected to VSS and MSEL0 pin float after chip reset. 

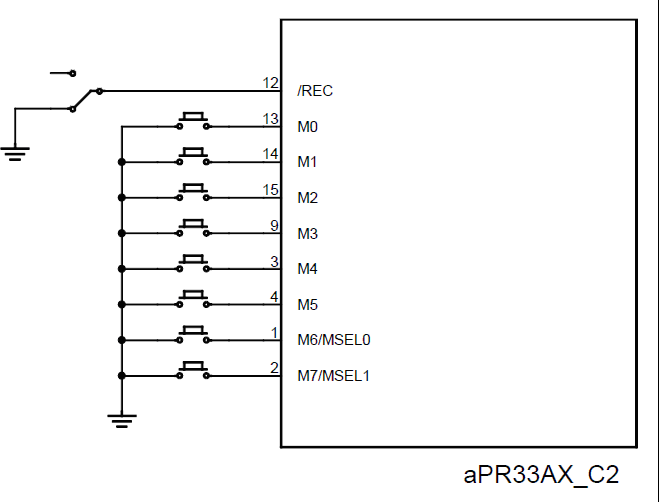
1-Message Mode

The memory will be for 1 message when both MSEL0 and MSEL1 pin connected to VSS after chip reset.



# RECORD MESSAGE

* During the /REC pin drove to VIL, chip in the record mode.
* When the message pin (M0, M1, M2 … M7) drove to VIL in record mode, the chip will playback “beep” tone and message record starting.
* The message record will continue until message pin released or full of this message, and the chip will playback “beep” tone 2 times to indicate the message record finished.
* If the message already exist and user record again, the old one’s message will be replaced.
* The following fig. showed a typical record circuit for 8-message mode. We connected a slide- switch between /REC pin and VSS, and connected 8 tact-switches between M0 ~ M7 pin and VSS. When the slide-switch fixed in VSS side and any tact-switch will be pressed, chip will start message record
* and until the user releases the tact-switch.
* Note: After reset, /REC and M0 to M7 pin will be pull-up to VDD by internal resistor.



# PLAYBACK MESSAGE

* During the /REC pin drove to VIH, chip in the playback mode.
* When the message pin (M0, M1, M2 … M7) drove from VIH to VIL in playback mode, the message playback starting.
* The message playback will continue until message pin drove from VIH to VIL again or end of this message.
* The following fig. showed a typical playback circuit for 8-message mode. We connected a slide-switch between /REC and VSS, and connected 8 tact-switches between M0 ~ M7 and VSS.
* When the slide-switch fixed in float side and any tact-switch will be pressed, chip will start message playback and until the user pressed the tact-switch again or end of message.
* Note: After reset, /REC and M0 to M7 pin will be pull-up to VDD by internal resistor.

# CHAPTER 4: SOFTWARE DESCRIPTION

This project is implemented using following software’s:

* Express PCB – for designing circuit
* Arduino IDE Studio compiler - for compilation part
* Proteus 7 (Embedded C) – for simulation part

### Express PCB:

Breadboards are great for prototyping equipment as it allows great flexibility to modify a design when needed; however the final product of a project, ideally should have a neat PCB, few cables, and survive a shake test. Not only is a proper PCB neater but it is also more durable as there are no cables which can yank loose.

Express PCB is a software tool to design PCBs specifically for manufacture by the company Express PCB (no other PCB maker accepts Express PCB files). It is very easy to use, but it does have several limitations.

It can be likened to more of a toy then a professional CAD program. It has a poor part library (which we can work around)

It cannot import or export files in different formats

It cannot be used to make prepare boards for DIY production

Express PCB has been used to design many PCBs (some layered and with surface-mount parts. Print out PCB patterns and use the toner transfer method with an Etch Resistant Pen to make boards. However, Express PCB does not have a nice print layout. Here is the procedure to design in Express PCB and clean up the patterns so they print nicely.

### Preparing Express PCB for First Use:

Express PCB comes with a less then exciting list of parts. So before any project is started head over to Audiologica and grab the additional parts by morsel, ppl, and tangent, and extract them into your Express PCB directory. At this point start the program and get ready to setup the workspace to suit your style.

Click View -> Options. In this menu, setup the units for “mm” or “in” depending on how you think, and click “see through the top copper layer” at the bottom. The standard color scheme of red and green is generally used but it is not as pleasing as red and blue.

### The Interface:

When a project is first started you will be greeted with a yellow outline. This yellow outline is the dimension of the PCB. Typically after positioning of parts and traces, move them to their final position and then crop the PCB to the correct size. However, in designing a board with a certain size constraint, crop the PCB to the correct size before starting.

Fig: 4.1 shows the toolbar in which the each button has the following functions:



### Fig 4.1: Tool bar necessary for the interface

* + - * The select tool: It is fairly obvious what this does. It allows you to move and manipulate parts. When this tool is selected the top toolbar will show buttons to move traces to the top / bottom copper layer, and rotate buttons.
      * The zoom to selection tool: does just that.
      * The place pad: button allows you to place small soldier pads which are useful for board connections or if a part is not in the part library but the part dimensions are available. When this tool is selected the top toolbar will give you a large selection of round holes, square holes and surface mount pads.
      * The place component: tool allows you to select a component from the top toolbar and then by clicking in the workspace places that component in the orientation chosen using the buttons next to the component list. The components can always be rotated afterwards with the select tool if the orientation is wrong.
      * The place trace: tool allows you to place a solid trace on the board of varying thicknesses. The top toolbar allows you to select the top or bottom layer to place the trace on.

### Design Considerations:

Before starting a project there are several ways to design a PCB and one must be chosen to suit the project’s needs.

Single sided, or double sided?

When making a PCB you have the option of making a single sided board, or a double sided board. Single sided boards are cheaper to produce and easier to etch, but much harder to design for large projects. If a lot of parts are being used in a small space it may be difficult to make a single sided board without jumpering over traces with a cable. While there’s technically nothing wrong with this, it should be avoided if the signal travelling over the traces is sensitive (e.g. audio signals).

A double sided board is more expensive to produce professionally, more difficult to etch on a DIY board, but makes the layout of components a lot smaller and easier. It should be noted that if a trace is running on the top layer, check with the components to make sure you can get to its pins with a soldering iron. Large capacitors, relays, and similar parts which don’t have axial leads can NOT have traces on top unless boards are plated professionally.

Ground-plane or other special purposes for one side?

When using a double sided board you must consider which traces should be on what side of the board. Generally, put power traces on the top of the board, jumping only to the bottom if a part cannot be soldiered onto the top plane (like a relay), and vice- versa.

Some projects like power supplies or amps can benefit from having a solid plane to use for ground. In power supplies this can reduce noise, and in amps it minimizes the distance between parts and their ground connections, and keeps the ground signal as simple as possible.

* 1. **Arduino IDE Compiler:**

This instructable adds to any of the Arduino on a Breadboard instructables.

1. We need a microcontroller with a pre-loaded Bootloader, or must load your own
2. Not all ATmega328’s are equal

(A bootloader, very simply, is a programme that sits on the chip and manages the upload of your sketches onto the chip)

### Procedural steps for compilation, simulation and dumping:

* + 1. **Compilation and simulation steps:**

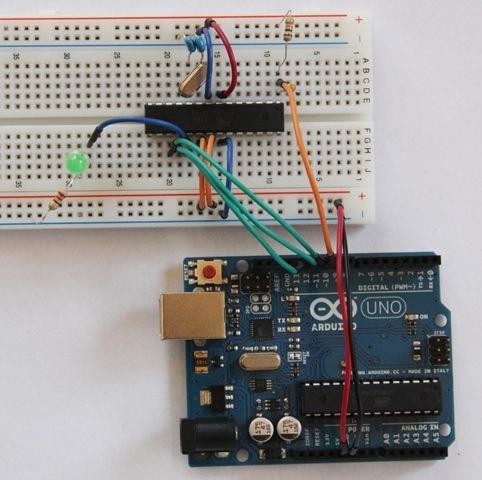
**Step 1: Parts**

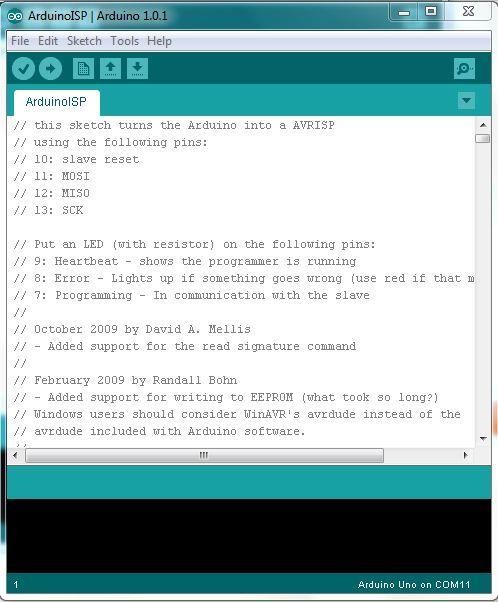
1 x Arduino on a Breadboard 1 x Arduino UNO Connecting Wires

Arduino IDE installed on your PC

**Step 2: The Approach**

We use the Arduino UNO to bootload the ATmega328 that is sitting on the Arduino-on-a-Breadboard. This is fairly straightforward having an ATmega328P-PU, but needs an extra step for an ATmega328- PU.



Step 3: Program your Arduino UNO as an ISP

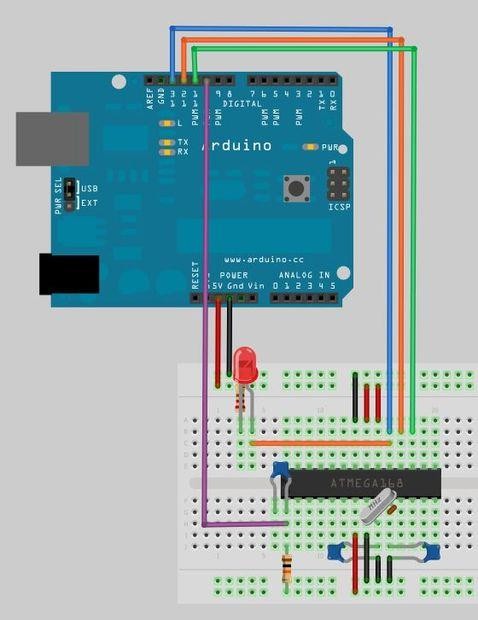
We need to program the Arduino UNO to act as an ISP (In-System Programmer), so that it can burn the bootloader onto the Breadboard chip.

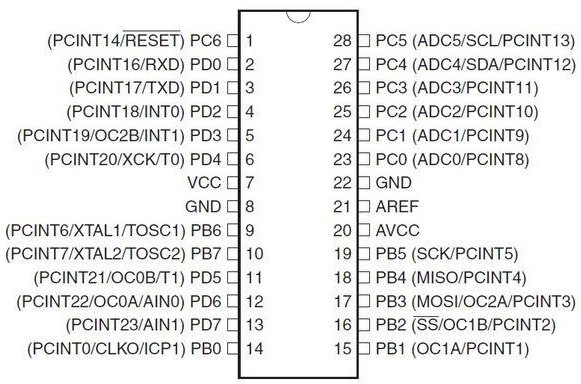
1. Open the Arduino IDE
2. Open the ArduinoISP sketch (under File, Examples)
3. If you’re using version 1.0 of the IDE:

Search for *void heartbeat* and change the line that reads:

*delay(40);* to *delay(20);*

Connect your UNO to the PC, making sure it’s not connected to the Arduino on a Breadboard. Ensure your UNO is selected under the Boards menu option, and upload the sketch.

Step 4: Connect your ATmega328



Now connect your ATmega to your UNO as follows:

* + UNO 5v ---> ATmega pin 7 (VCC)
  + UNO GND ---> ATmega pin 8 (GND)
  + UNO pin 10 ---> ATmega pin 1 (RESET)
  + UNO pin 11 ---> ATmega pin 17 (MOSI)
  + UNO pin 12 ---> ATmega pin 18 (MISO)
  + UNO pin 13 ---> ATmega pin 19 (SCK)

Step 5: Which ATmega328 are you using?



The **-PU** suffix means that the chips are in a PDIP package, the format we need for our breadboard.

The **328P** is a [picoPower](http://www.atmel.com/technologies/lowpower/default.aspx) processor, designed for low power consumption, and is used on the Arduino boards. Given low power consumption this is first choice.

The **328** does *not* have picoPower technology, and is not used on the Arduino boards – and is not explicitly supported by the Arduino IDE.

What this means is that we can easily bootload the ATmega328P, but not the ATmega328. Unfortunately the websites that sell these chips don't always differentiate between them and forums are filled with people struggling to use the ATmega328-PU.

Luckily there is a workaround - take a look at my [Crash Bang website.](http://www.crash-bang.com/resource/bootload-atmega328/)

Step 6: ATmega328-PU workaround



Each microprocessor has a **signature** – a unique code that identifies its model. When you bootload a chip (or even upload a sketch) the Arduino IDE checks that the chip selected matches the type it’s connected to. Even though the ATmega328-PU in essence functions in the same way as the ATmega328P-PU, it has a different signature, and one that isn’t recognised by the Arduino IDE.

In your Arduino folder, find the subfolder: *..\hardware\tools\avr\etc*

1. Make a backup copy of the file: *avrdude.conf*
2. Open the file *avrdude.conf* in a text editor
3. Search for: “*0x1e 0x95 0x0F*” (this is the ATmega328P signature)
4. Replace it with: “*0x1e 0x95 0x14*” (this is the ATmega328 signature)
5. Save the file
6. Restart the Arduino IDE
7. Continue with the rest of the steps in the instructable, and once bootloading is complete restore the backup copy you made.



Step 7: Bootload the ATmega328



**In the Arduino IDE,** from the *Tools* menu:

* + under the *Board* option choose *Arduino UNO*
  + under the *Serial Port* option ensure the correct port is selected
  + under the *Programmer* option choose *Arduino as ISP*

**To burn the Bootloader**, choose *Burn Bootloader* from the *Tools* menu

You should see a message *“Burning bootloader to I/O Board (this may take a minute)"*

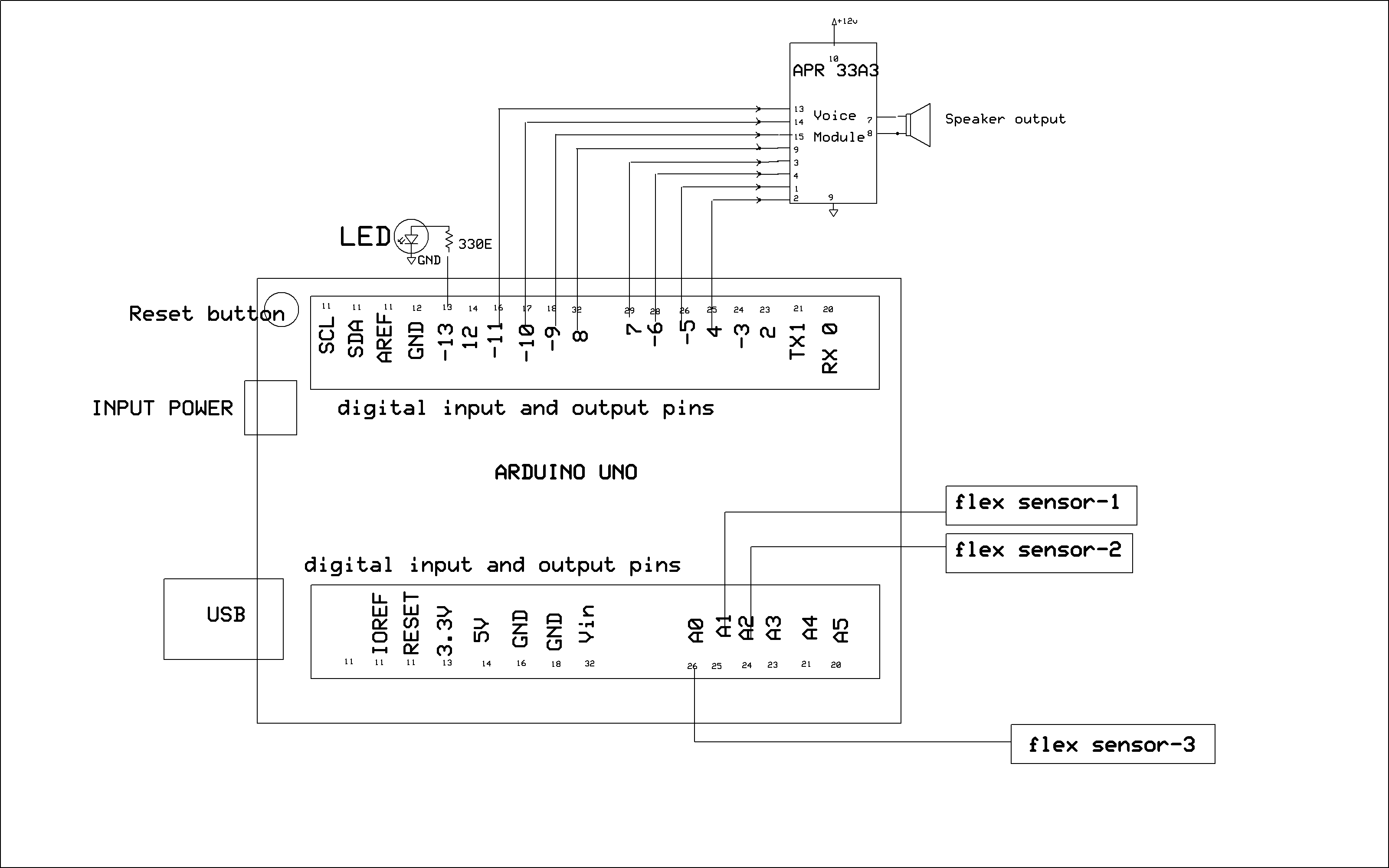
Once the bootloader has been burned, a message of confirming the success gets displayed.

”**Congratulations: You're now ready to load sketches onto your Arduino on a breadboard!”**

# CHAPTER 5: PROJECT DESCRIPTION

In this chapter, schematic diagram and interfacing of arduino with each module is

considered**.**



The above schematic diagram of **Smart Gloves for Dumb and Deaf** explains the interfacing section of each component with micro controller and flex sensor, voice module.

# CHAPTER 6: ADVANTAGES AND DISADVANTAGES

### Advantages:

1. It requires fewer components so its cost is low.
2. Small in size: Due to small size we can place its hardware on our hand easily.
3. Light weight.
4. Flexible to users.
5. Easy to operate: Anyone can operate it easily.
6. Real time translation.
7. Low power consumption.

### Disadvantages:

1. Limited distance.

### Applications:

* + It is used for dumb people mainly.
  + Automatically turn off the devices by gestures.

# CHAPTER 7: RESULTS

### Result:

The project “**Smart Gloves for Dumb and Deaf**” was designed a smart glove which is different voices can be announced by folding flex sensors.

### Conclusion:

Integrating features of all the hardware components used have been developed in it. Presence of every module has been reasoned out and placed carefully, thus contributing to the best working of the unit. Secondly, using highly advanced IC’s with the help of growing technology, the project has been successfully implemented. Thus the project has been successfully designed and tested.

### Future Scope:

* + - We can add device control to this project.

# REFERENCES:

* + - 1. Hussana Johar R.B, Priyanka A, Revathi Amrut M S, Suchitha K, Sumana K J “Multiple sign language translation into voice” International Journal of Engineering and Innovative Technology (IJEIT), Volume 3, Issue 10, April 2014 .
      2. Cao Dong, Ming C.Leu, Zhaozheng Yin, “ American sign language Alphabet Recognition Using Microsoft Kinect”, Computer Vision and pattern Recognition workshop, IEEE conference, pp: ,2015.
      3. Nicholas Born.,”Senior Project Sign Language Glove”, electrical engineering department. California Polytechnic State University, 1- 49,2010
      4. Kirsten Ellis and Jan Carlo Barca.”Exploring Sensor Gloves for Teaching Children Sign Language. Advances in Human Computer Interaction”.1-8., 2012
      5. SolankiKumar,“ Indian Sign Language using Flex sensor Glove” International Journal of Engineering Trends and Technology(IJETT)vol.4,n0.6 June 2013.