CHAPTER 1: COMMUNICATION DEVICE FOR MUTE PEOPLE

1.1 INTRODUCTION:

In general, embedded system is a combination of both hardware and software used to achieve a unified single task/performance. Importantly it monitors, responds to/or control an external environment that is well connected to a system through sensors, actuators and other interfaces and it meets timing and other constraints imposed on it by the environment. However, embedded systems require a processor that is an important unit and mostly like an heart. The processing core in the embedded system is either a microcontroller or digital signal processors (DSP).

In the modern society, there is a consistent increase in the number of people who suffer from various illness that includes metabolic and life style related diseases, though most of the infectious diseases are under control and cured. This condition always looks for better diagnostic tools, expedited information transfer technologies, accurate prognostic devices, importantly patient management system and other alternative medicine strategies. Recently, embedded system technologies have occupied an important area in the biomedical application in diagnosis, prognosis, patient management and telemedicine that broadly includes the diseases of various kinds such as infectious, metabolic and lifestyle related disease in human beings.

It is capable of converting the hand gesture into codes that are understandable by computer. However these codes are routed to particular letter and then they are displayed in LCD monitor for human to understand. Each letter is displayed in LCD one by one, and so it takes time to understand

as a word and then to respond back. Number of codes can be generated is lesser as it does not use a MEMS based Accelerometer.

1.2 DRAWBACKS:

- Difficult for illiterates.
- Only lesser number of codes can be generated.
- Did not use MEMS.

In Our proposed system, pre recorded is played for each hand gesture and we use MEMS for accurate gesture recognition. Thus, the drawbacks of the existing system is overcome by our proposed system and it is briefly described in the chapter 3.

CHAPTER 2 : LITERATURE SURVEY

Our project is developed based on the ideas drawn from the following papers.

In reference [1], the sign language is converted into voice using the concepts of neural networks, hidden Markov models, natural language processing, linguistics and parallel computing which makes this system complicated.

In reference [2], voice encryption is done using JADE algorithm. Thus, the output produced in the form of computer generated voice which makes the user uncomfortable.

In reference [3], in order to communicate a single word in the form of voice, each letter in that word has to be expressed in gesture which makes the communication inefficient and time consuming.

In reference [4], only flex sensors are used to detect the hand gesture. Hence, large number of signs cannot be expressed which limits the communication level.

In reference [5], the hand gesture has to be done for every alphabet which is converted into its corresponding ASCII code and it is displayed in the form of letter using LCD. Since, the output is in the form of text the illiterate and the physically challenged people find it difficult to communicate with the mute people.

In reference [6], the sign language is converted into text and it can be displayed only in devices which supports Bluetooth. It makes the communication difficult if the Bluetooth supporting devices are not available**.**

CHAPTER 3: PROPOSED SYSTEM

3.1 INTRODUCTION :

Since the output of the existing system is in text form illiterate people find it difficult to understand. Hence in the proposed system, the digital value generated by ADC is mapped to recorded voice. Therefore when mute peoples show their gesture, the pre recorded voice is played. Using this system, the communication between the mute people and others is made easy..

This chapter consists of the hardware and software requirements, which is used in the block diagram. Flow of operation is shown for the betterment of understanding.

3.2 HARDWARE REQUIREMENTS :

- Flex Sensors – 3 nos.
- MEMS based Accelerometer
- MSP430 Microcontroller
- Zigbee module
- APR9600 - Integrated Chip

3.3 SOFTWARE REQUIREMENTS :

- Energia (Open Source Software)

3.4 BLOCK DIAGRAM :

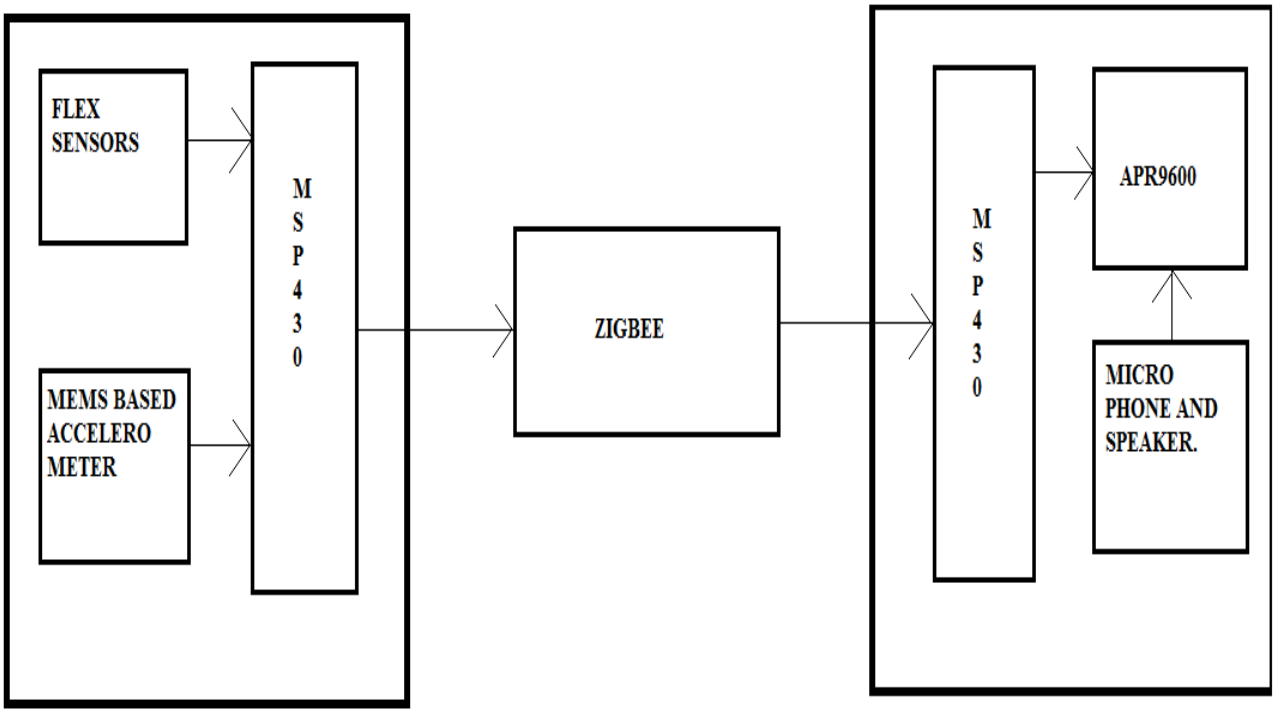


Fig No:3.1.Block Diagram Of Proposed System.

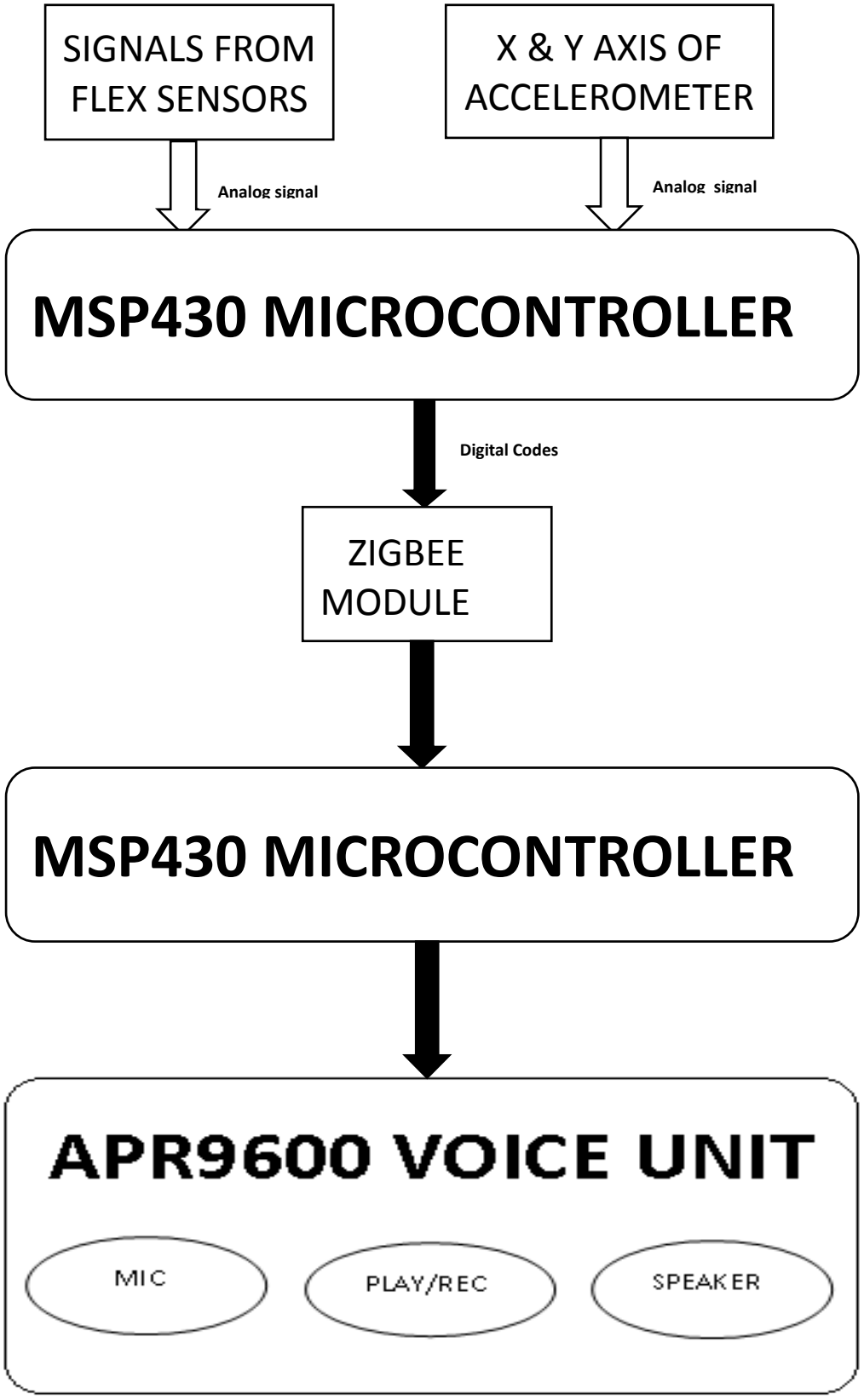
3.5 EXPLANATION :

The block diagram consists of two major units – Detection unit and Voice unit. The signals from detection unit are transmitted to the voice unit using ZIGBEE Protocol.

The three signals from Flex sensors and two signals from accelerometer are detected using MSP430 controller. Using ADC, those analog signals are converted to digital signals and are transmitted to the voice unit using Zigbee module.

The voice unit which contains APR9600 IC receives those digital signals and plays back its corresponding recorded audio using the speaker.

3.6 FLOW CHART :



CHAPTER 4: HARDWARE EXPLANATION

4.1. DETECTION UNIT :

- The detection unit consists of three blocks-

a) Flex Sensors

b) Accelerometer (based on MEMS)

c) MSP430 (microcontroller).

Sensors:

A sensor is a transducer whose purpose is to sense (that is, to detect) some characteristic of its environs. It detects events or changes in quantities and provides a corresponding output, generally as an electrical or optical signal; for example, a thermocouple converts temperature to an output voltage. But a mercury-in-glass thermometer is also a sensor; it converts the measured temperature into expansion and contraction of a liquid which can be read on a calibrated glass tube.

4.1.1. FLEX SENSORS :

The flex sensor patented technology is based on resistive carbon elements. They require a 5volt input and output between 0 and 5 volt, the resistivity varying with the sensor's degree of bend and the voltage output changing accordingly. It will only change resistance in 1 direction. An unflexed sensor has a resistance of about 10,000 ohms. As the flex sensor is bent, the resistance increases to 30-40 kilo ohms at 90 degrees.

4.1.1.1. MECHANICAL SPECIFICATIONS :

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

4.1.1.2.ELECTRICAL SPECIFICATIONS:

- Flat Resistance: 10K Ohms ±30%
- Bend Resistance: minimum 2 times greater than the flat resistance at 180° pinch bend .
- Power Rating : 0.5 Watts continuous; 1 Watt peak.

Fig 4.1.1: A Glove with Flex Sensors

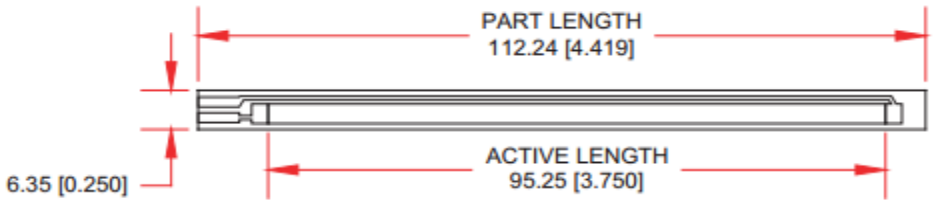


Fig 4.1.2: Dimensional Diagram

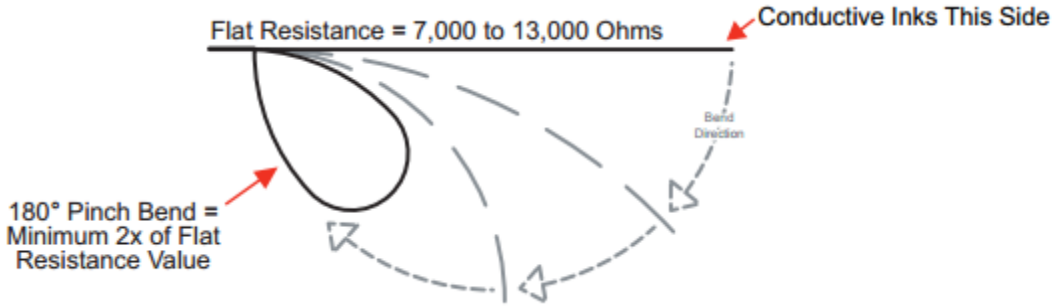


Fig 4.1.3: Basic Flex Sensor Circuit

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 error due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324.

4.1.1.3. 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.

4.1.1.4. ADJUSTABLE BUFFER :

A potentiometer can be added to the circuit to adjust the sensitivity range.

4.1.1.5. 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.

4.1.1.6. FEATURES :

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
    1. Robotics
    2. Gaming (Virtual Motion)
    3. Medical Devices
    4. Computer Peripherals
    5. Musical Instruments
    6. Physical Therapy
- Simple Construction
- Low Profile

4.1.2. MEMS BASED ACCELEROMETER :

The MMA7361L is a low power, low profile capacitive micro machined accelerometer featuring signal conditioning, a 1-pole low pass filter, temperature compensation, self test. It provides low voltage operation ranging from 2.2v to 3.6v. It has a robust design and high shock survivability. It a low cost device.

4.1.2.1. DESCRIPTION :

This triaxial accelerometer combines three orthogonally mounted SDI integrated low noise accelerometers in a rugged case for measuring accelerations in commercial/industrial environments.

This module is tailored for zero to medium frequency instrumentation applications. Its anodized alluminium case is epoxy sealed via two #8 screws(or m4).

On board voltage regulation and an internal voltage reference eliminate the need for precision power supplies. It is relatively insensitive to temperature changes and gradients.

4.1.2.2. OPERATION :

This accelerometer produces three differential analog output voltage pairs which vary with accelerations. The signal outputs are fully differential about a common mode voltage of approximately 2.5V

The output scale factor is independent from the supply voltage of +8 to +32V. At zero acceleration the output differential voltages nominally zero volts DC. At (+ or -) full scale acceleration output is (+ or -)4V DC respectively. The axis directions are marked on the case with positive accelerations defined as acceleration in the direction of axis arrow.

Fig 4.1.4: Triaxial Accelerometer.

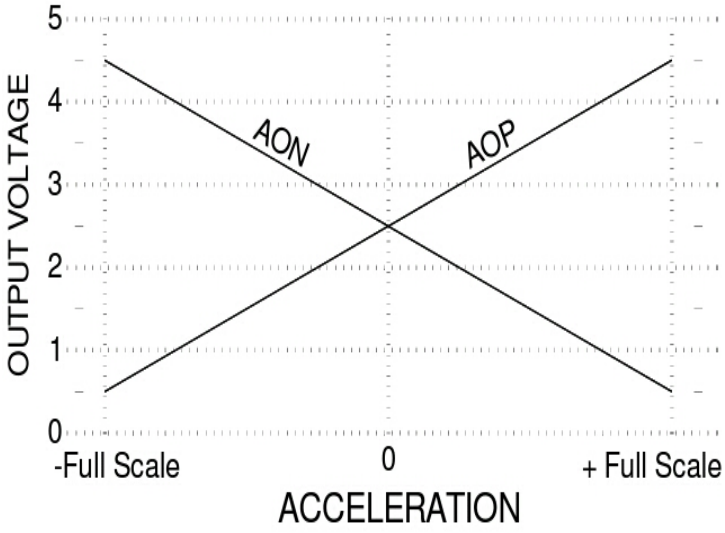


FIG 4.1.5: OUTPUT VOLTAGE VS ACCELERATION GRAPH.

4.1.2.3. FEATURES:

- Three axis acceleration sensing.
- capacitive micro machined.
- nitrogen damped.

- (+/-)4volt differential output or 0.5 volt to 4.5 volt single ended output.
- fully calibrated.
- low power consumption.
- good noise performance.

4.1.2.4. APPLICATIONS :

- Vibration monitoring and analysis.
- Machine contol
- Modal analysis
- Robotics
- Crash testing
- Instrumentation
- Rotating machinery control

4.1.3. MSP430 MICROCONTROLLER:

The MSP430 MCU is designed specifically for ultra-low-power applications. Its flexible clocking system, multiple low-power modes, instant wakeup and intelligent autonomous peripherals enable true ultra-low-power optimization, dramatically extending battery life. MSP430 MCUs are highly integrated and offer a wide range of high performance analog and digital peripherals

4.1.3.1. DESCRIPTION:

Texas Instruments , a pioneer in low power Microcontroller market has released the MSP 430 Microcontroller in the late 1990s.This is a 16-bit, RISC- based, mixed-signal processor. MSP430 MCUs have the right mix of intelligent peripherals , ease-of-use, low cost and lowest power consumption for thousands of applications

Typical applications include embedded and sensor systems. Integrated timers make the configurations ideal for industrial control applications such as ripple counters, digital motor control, EE-meters, hand-held meters, etc. The hardware multiplier enhances the performance and offers a broad code and hardware-compatible family solution.

Ultra-low-power (ULP) architecture and flexible clock system extend battery life. Low power consumption: 0.1 μ A for RAM data Retention, 0.8 μ A for RTC mode operation 250 μA /MIPS at active operation. Low operation voltage (from 1.8 V to 3.6 V).

Zero-power Brown-Out Reset (BOR). Enhanced libraries to benefit several applications such as capacitive touch, metering metrology, low power design and debugging. Extensive interrupt capability relieves need for polling. Flexible and powerful processing capabilities: Seven source-address modes.

Four destination-address modes. Only 27 core instructions. Prioritized, nested interrupts. Large register file. Efficient table processing. Fast hex-to-decimal conversion

The MSP430 CPU has a 16-bit RISC architecture. The architecture of MSP430 Can be understood from the following diagram. The controlles performance is directly related to the 16-bit data bus, the 7 addressing modes and the reduced instructions set , which allows a shorter, denser programming code for fast execution.

These MSP controller families share a 16-bit CPU core, RISC type, intelligent peripherals , and flexible clock system that interconnects using a Von Neumann common memory address bus (MAB) and memory data bus (MDB) architecture.

The CPU of MSP 430 includes a 16-bit ALU and a set of 16 Registers R0 –R15.In these registers Four are special Purpose and 12 are general purpose registers . All the registers can be addressed in the same way. The special Purpose Registers are PC (Program Counter), SP (Stack Pointer) , SR (Status Register) and CGx (Constant Generator)

All memory, including RAM, Flash/ROM, information memory, special function registers (SFRs), and peripheral registers are mapped into a single, contiguous address space. The CPU is capable of addressing data values either as bytes (8 bits) or words (16 bits). Words are always addressed at an even address , which contain the least significant byte, followed by the next odd address, which contains the most significant byte.

For 8-bit operations, the data can be accessed from either odd or even addresses, but for 16-bit operations, the data values can only be accessed from even addresses.

The interrupt vector table is mapped at the very end of memory space (upper 16 words of Flash/ROM), in locations 0FFE0h through to 0FFFEh (see the device-specific datasheets). The priority of the interrupt vector increases with the word address . The start address of Flash/ROM depends on the amount of Flash/ROM present on the device. The start address varies between01100h (60k devices) to 0F800h (2k devices) and always runs to the end of the address space at location 0FFFFh.

Flash can be used for both code and data. Word or byte tables can also be stored andread by the program from Flash/ROM. All code, tables, and hard-coded constants reside in this memory Space. The MSP430 flash devices contain an address space for information memory. It is like an onboard EEPROM, where variables needed fort he next power up can be stored during power down. It can also be used as code memory.

The MSP430 flash devices contain an address space for boot memory, located between addresses 0C00h through to 0FFFh. The“ bootstrap loader” is located in this memory space, which is an External interface that can be used to program the flash memory in addition to the JTAG. This memory region is not accessible by other applications, so it cannot be overwritten accidentally.

RAM always starts at address 0200h. The end address of RAM depends on the amount of RAM present on the device. RAM is used for both code and data. Peripheral modules consist of all on-chip peripheral registers that are mapped into the address space. These modules can be accessed with byte or word instructions, depending if the peripheral module is 8-bit or 16-bit respectively. The 16-bit peripheral modules are located in the address space from addresses 0100 through to 01FFh and the 8-bit peripheral modules are mapped into memory from addresses 0010h through to 00FFh.

The Special Function Registers (SFRs) are located at memory addresses from 0000h to 000Fh. SFRs must be accessed using byte instructions only.

The CPU has a 16-bit ALU, four dedicated registers and twelve working registers, which makes the MSP430 a high performance microcontroller suitable for low power applications. The addition of twelve working general purpose registers saves CPU cycles by allowing the storage of frequently used values and variables instead of using RAM.

The MSP430 CPU includes an arithmetic logic unit (ALU) that handles addition, subtraction, comparison and logical (AND, OR, XOR) operations. ALU operations can affect the overflow, zero, negative, and carry flags in the status register.R0: Program Counter (PC) . The 16-bit Program Counter (PC/R0) points to the next instruction to be read from memory and executed by the CPU. The Program counter is implemented by the number of bytes used by the instruction (2, 4, or 6 bytes, always even). It is important to note that the PC is aligned at even addresses, because the instructions are 16 bits, even though the individual memory addresses contain 8-bit value.

The Stack Pointer (SP/R1) is located in R1.Stack can be used by user to store data for later use(instructions : store by PUSH, retrieve by POP). Stack can be used by user or by compiler for subroutine parameters

Stack can be used by subroutine calls to store the program counter value for return at subroutines end (RET). used by interrupt - system stores the actual PC value first, then the actual status register content (on top of stack) on return from interrupt (RETI) the system get the same status as just before the interrupt happened (as long as none has changed the value on TOS) and the same program counter value from stack.

The Status Register (SR/R2) is a 16 bit register , andit stores the state and control bits. The system flagsare changed automatically by the CPU depending onthe result of an operation in a register. The reservedbits of the SR are used to support the constants generator.

Depending of the source-register addressing modes (As) value, six commonly used constants can be generated without a code word or code memory access to retrieve them. This is a very powerful feature, which allows the implementation of emulated instructions, for example, instead of implementing a core instruction for an increment, the constant generator is used.

These general-purpose registers are used to store data values , address pointers, or index values and can be accessed with byte or word instructions.

The MSP430 supports seven addressing modes for the source operand and four addressing modes for the destination operand . They are Register mode, Indexed mode, Symbolic mode, Absolute mode, Indirect register mode, Indirect auto increment mode and Immediate mode.

Register mode operations work directly on the processor registers, R4 through R15, or on special function registers, such as the program counter or status register. They are very efficient in terms of both instruction speed and code space.Ex : MOV R4, R5 Move (copy) the contents of source (register R4) to destination (register R5). Register R4 is not affected.

The Indexed mode commands are formatted as X(Rn), where X is a constant and Rn is one of the CPU registers. The absolute memory location X+Rn is addressed. Indexed mode addressing is useful for applications such as lookup tables Ex : MOV F000h(R5), R4 Move (copy) the contents at source address (F000h +R5) to destination (register R4)

Symbolic mode allows the assignment of labels to fixed memory locations, so that those locations can be addressed. This is useful for the development of embedded programs. MOV XPT, YPT Move the content of source address XPT (x pointer) to the destination address YPT (y pointer).

Similar to Symbolic mode, with the difference that the label is preceded by "&". The word following the instruction contains the absolute address. X is stored in the next word. Indexed mode X(SR) is used MOV &XPT, &YPT Move the content of source address XPT to the destination address YP.

The data word addressed is located in the memory location pointed to by Rn. Indirect mode is not valid for destination operands, but can be emulated with the indexed mode format 0(Rn). Here Rn is used as a pointer to the operand. MOV @(R4), R5 Move the contents of the source address (contents of R4) to the destination (register R5). Register R4 is not modified.

Similar to indirect register mode, but with indirect auto increment mode, the operand is incremented as part of the instruction. The format for operands is @Rn+. This is useful for working on blocks of data. Rn is used as a pointer to the operand. Rn is incremented afterwards by 1 for byte instructions and by 2 for word instructions. Ex: MOV @R4+, R5 Move the contents of the source address (contents of R4) to the destination (register R5), then increment the value in register R4 to point to the next word.

Immediate mode is used to assign constant values to registers or memory locations.MOV #E2h, R5 Move the immediate constant E2h to the destination (register R5).

The MSP430 instruction set consists of 27 core instructions. Additionally, it supports 24 emulated instructions. The core instructions have unique op-codes decoded by the CPU, while the emulated ones need assemblers and compilers to generate their mnemonics.

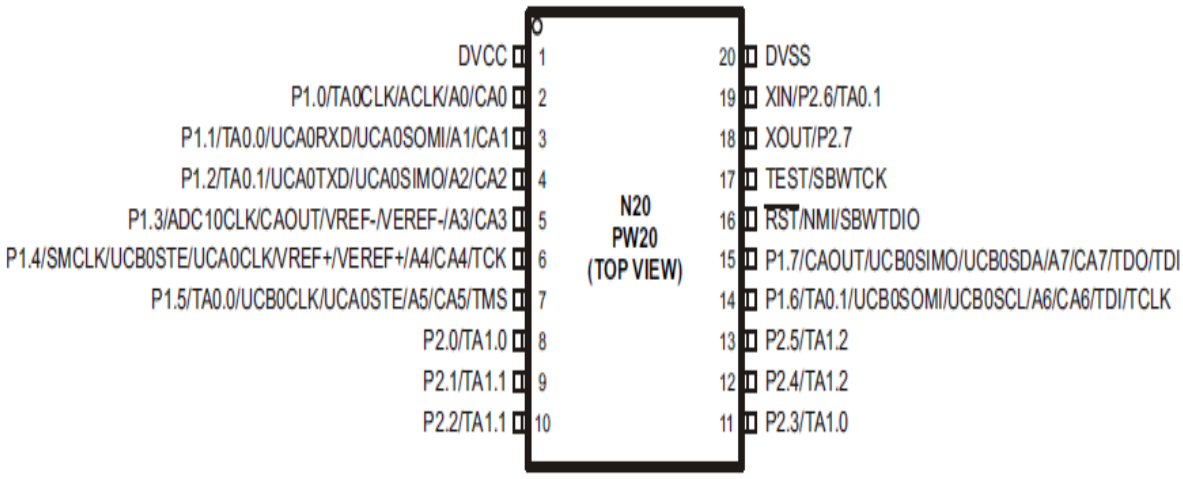


Fig 4.1.6:Device Pinout,Msp430G2x13 And Msp430G2x53, 20-Pin Devices, TSSOP And PDIP

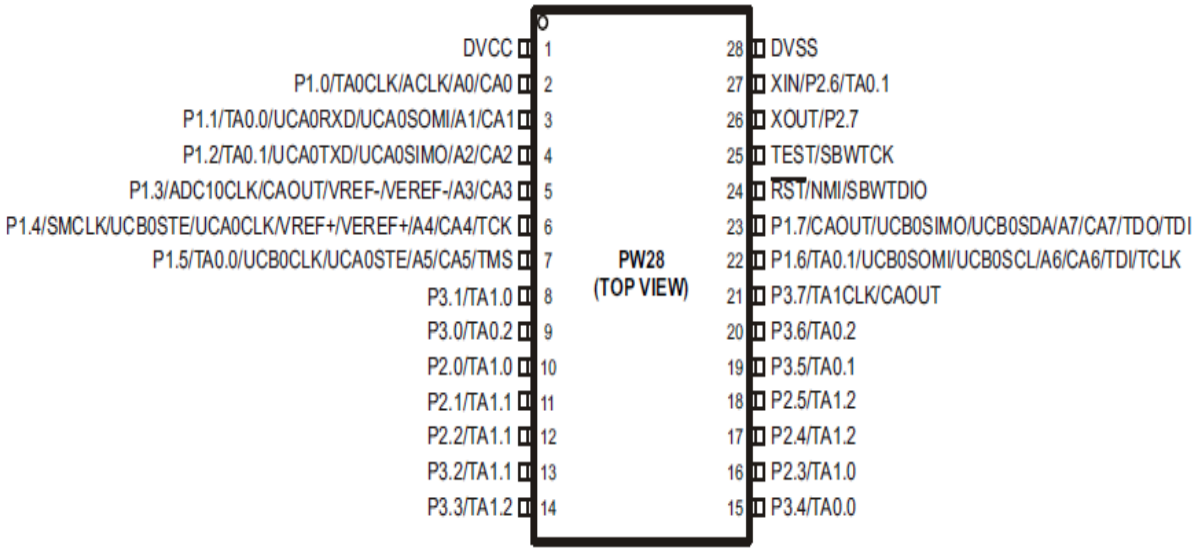


Fig4.1.7:Device Pinout,Msp430g2x13 And Msp430g2x53, 28-Pin Devices, Ts

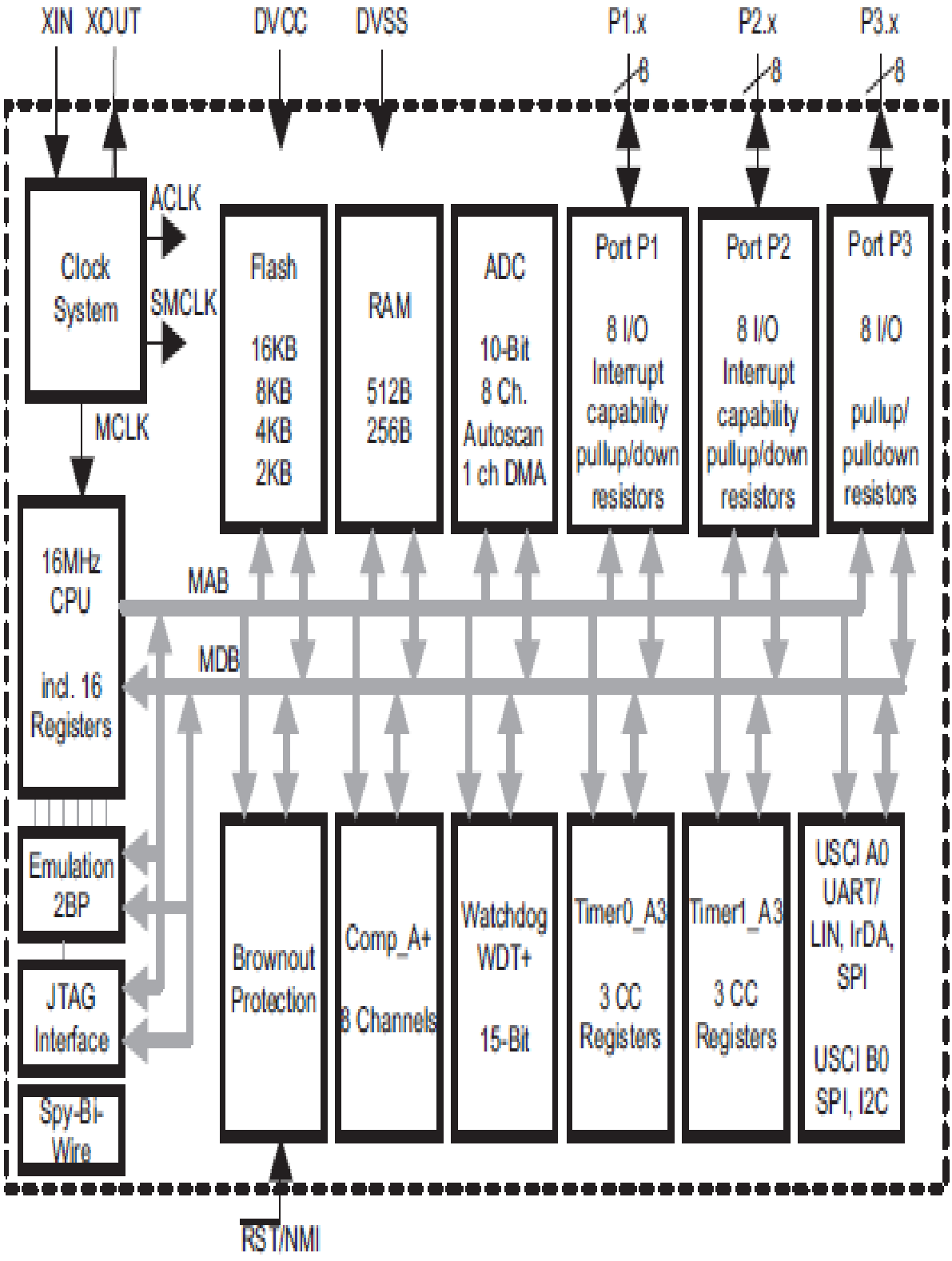


Fig No:4.1.8. Functional Block Diagram Of Msp430G2x53

TERMINAL				I/O	DESCRIPTION
NAME	NO				
	PW20,N20	PW28	RHB32		
P1.0/ TA0CLK/ ACLK/ A0 CA0	2	2	31	I/O	General-purpose digital I/O pin Timer0_A, clock signal TACLK input ACLK signal output ADC10 analog input A0(1) Comparator_A+, CA0 input
P1.1/ TA0.0/ UCA0RXD/ UCA0SOMI/ A1/ CA1	3	3	1	I/O	General-purpose digital I/O pin Timer0_A, capture: CCI0A input, compare: Out0 output / BSL transmit USCI_A0 UART mode: receive data input USCI_A0 SPI m5ode: slave data out/master in ADC10 analog input A1(1) Comparator_A+, CA1 input
P1.2/ TA0.1/ UCA0TXD/ UCA0SIMO/ A2/ CA2	4	4	2	I/O	General-purpose digital I/O pin Timer0_A, capture: CCI1A input, compare: Out1 output USCI_A0 UART mode: transmit data output USCI_A0 SPI mode: slave

					data in/master out ADC10 analog input A2(1) Comparator_A+, CA2 input
P1.3/ ADC10CLK/ A3/ VREF-/VEREF-/ CA3/ CAOUT	5	5	3	I/O	General-purpose digital I/O pin ADC10, conversion clock output(1 ADC10 analog input A3(1) ADC10 negative reference voltage (1) Comparator_A+, CA3 input Comparator_A+, output
P1.4/ SMCLK/ UCB0STE/ UCA0CLK/ A4/ VREF+/VEREF+ / CA4/ TCK	6	6	4	I/O	General-purpose digital I/O pin SMCLK signal output USCI_B0 slave transmit enable USCI_A0 clock input/output ADC10 analog input A4(1) +/ ADC10 positive reference voltage Comparator_A+, CA4 input JTAG test clock, input terminal for device programming and test
P1.5/ TA0.0/	7	7	5	I/O	General-purpose digital I/O pin

UCB0CLK/ UCA0STE/ A5/ CA5/ TMS					Timer0_A, compare: Out0 output / BSL receive USCI_B0 clock input/output O USCI_A0 slave transmit enable ADC10 analog input A5(1) Comparator_A+, CA5 input JTAG test mode select, input terminal for device programming and test
P1.6/ TA0.1/ A6/ CA6/ UCB0SOMI/ UCB0SCL/ TDI/TCLK	14	22	21	I/O	General-purpose digital I/O pin / Timer0_A, compare: Out1 output / ADC10 analog input A6(1) Comparator_A+, CA6 input USCI_B0 SPI mode: slave out master in USCI_B0 I2C mode: SCL I2C clock JTAG test data input or test clock input during programming and test.
P1.7/ A7/ CA7/ CAOUT/	15	23	22	I/O	General-purpose digital I/O pin ADC10 analog input A7(1) Comparator_A+, CA7 input

UCB0SIMO/ UCB0SDA/ TDO/TDI					Comparator_A+, output / USCI_B0 SPI mode: slave in master out USCI_B0 I2C mode: SDA I2C data JTAG test data output terminal or test data input during programming and test(2)
P2.0/ TA1.0	8	10	9	I/O	General-purpose digital I/O pin Timer1_A, capture: CCI0A input, compare: Out0 output
P2.1/ TA1.1	9	11	10	I/O	General-purpose digital I/O pin Timer1_A, capture: CCI1A input, compare: Out1 output
P2.2/ TA1.1	10	12	11	I/O	General-purpose digital I/O pin Timer1_A, capture: CCI1B input, compare: Out1 output
P2.3/ TA1.0	11	16	15	I/O	General-purpose digital I/O pin Timer1_A, capture: CCI0B input, compare: Out0 output
P2.4/	12	17	16	I/O	General-purpose digital I/O

TA1.2					pin Timer1_A, capture: CCI2A input, compare: Out2 output
P2.5/ TA1.2	13	18	17	I/O	General-purpose digital I/O pin Timer1_A, capture: CCI2B input, compare: Out2 output
XIN/ P2.6/ TA0.1	19	27	26	I/O	Input terminal of crystal oscillator General-purpose digital I/O pin Timer0_A, compare: Out1 output
XOUT/ P2.7	18	26	25	I/O	Output terminal of crystal oscillator(3) General-purpose digital I/O pin
P3.0/ TA0.2	-	9	7	I/O	General-purpose digital I/O pin Timer0_A, capture: CCI2A input, compare: Out2 output
P3.1/ TA1.0	-	8	6	I/O	General-purpose digital I/O pin Timer1_A, compare: Out0 output
P3.2/ TA1.1	-	13	12	I/O	General-purpose digital I/O pin Timer1_A, compare: Out1

					output
P3.3/ TA1.2	-	14	13	I/O	General-purpose digital I/O pin Timer1_A, compare: Out2 output
P3.4/ TA0.0	-	15	14	I/O	General-purpose digital I/O pin Timer1_A, compare: Out0 output
P3.5/ TA0.1	-	19	18	I/O	General-purpose digital I/O Timer0_A, compare: Out1 output
P3.6/ TA0.2	-	20	19	I/O	General-purpose digital I/O Timer0_A, compare: Out2 output
P3.7/ TA1CLK	-	21	20	I/O	General-purpose digital I/O Timer1_A, clock signal TACLK input Comparator_A+, output
RST/ NMI/ SBWTDIO	16	24	23	I	Reset Nonmaskable interrupt input Spy-Bi-Wire test data input/output during programming and test
TEST/ SBWTCK	17	25	24	I	Selects test mode for JTAG pins on Port 1. The device protection fuse is connected to TEST.

					Spy-Bi-Wire test clock input during programming and test
AVCC	NA	NA	29	NA	Analog supply voltage
DVCC	1	1	30	NA	Digital supply voltage
DVSS	20	28	27,28	NA	Ground reference
NC	NA	NA	8,32	NA	Not connected
Not connected	NA	NA	PAD	NA	QFN package pad. Connection to VSS is recommended.

Table No:4.1.1. Terrminal Functions

4.1.3.2. FEATURES:

- Low Supply-Voltage Range: 1.8 V to 3.6 V. Universal Serial Communication Interface
- Ultra-Low Power Consumption (USCI)
    - Active Mode: 230 μA at 1 MHz, 2.2 V – Enhanced UART Supporting Auto Baudrate
    - Standby Mode: 0.5 μA Detection (LIN)
    - Off Mode (RAM Retention): 0.1 μA – IrDA Encoder and Decoder
- Five Power-Saving Modes – Synchronous SPI
- Ultra-Fast Wake-Up From Standby Mode in – I2C™
- Less Than 1 μs • On-Chip Comparator for Analog Signal
- 16-Bit RISC Architecture, 62.5-ns Instruction Compare Function or Slope Analog-to-Digital
- Cycle Time (A/D) Conversion

- Basic Clock Module Configurations • 10-Bit 200-ksps Analog-to-Digital (A/D)
- Converter With Internal Reference, Sample-
    - Internal Frequencies up to 16 MHz With and-Hold, and Autoscan
- Four Calibrated Frequency
- Brownout Detector
    - Internal Very-Low-Power Low-Frequency
- (LF) Oscillator • Serial Onboard Programming,
- No External Programming Voltage Needed,
    - 32-kHz Crystal Programmable Code Protection by Security
    - External Digital Clock Source Fuse
- Two 16-Bit Timer_A With Three • On-Chip Emulation Logic With Spy-Bi-Wire
- Capture/Compare Registers Interface
- Up to 24 Capacitive-Touch Enabled I/O Pins
- Package Options
    - TSSOP: 20 Pin, 28 Pin
        - PDIP: 20 Pin
        - QFN: 32 Pin.

4.2. ZIGBEE MODULE (IEEE 802.15.4):

Zigbee is used for wireless communication. It acts as a transceiver between the detection and voice unit.

4.2.1. WHAT ZIGBEE IS?

- Technological Standard Created for Control and Sensor Networks
- Based on the IEEE 802.15.4 Standard
- Created by the ZigBee Alliance

FEATURES	IEEE 802.11b	BLUETOOTH	ZIGBEE
Power profile	Hours	months	Years
Complexity	Very complex	complex	Simple
Nodes/master	32	7	6400
Latency	Enumeration upto 3 seconds	Enumeration upto 10 seconds	Enumeration 30ms
Range	100m	10m	70 to 300m
Extendability	Roaming possible	No	Yes
Data rate	11 Mbps	1Mbps	250 kbps
Security	Authentication service set ID (SSID)	64 bit, 128 bit	128 bit AES and application layer user defined

Table No 4.2.1 Comparing Zigbee With Other Wireless Technologies

4.2.2. WHY ZIGBEE ?

• Reliable

• Very long battery life

• Secure

• Low cost

• Can be used globally

4.2.3. IEEE 802.15.4 BASICS :

802.15.4 is a simple packet data protocol for lightweight wireless networks

- Channel Access is via Carrier Sense Multiple Access with collision avoidance and optional time slotting
- Message acknowledgement and an optional beacon structure –Multi-level security
- Three bands, 27 channels specified
    - •2.4 GHz: 16 channels, 250 kbps
    - •868.3 MHz: 1 channel, 20 kbps
    - •902-928 MHz: 10 channels, 40 kbps
- Works well for
    - •Long battery life, selectable latency for controllers, sensors, remote monitoring and portable electronics
- Configured for maximum battery life, has the potential to last as long as the shelf life of most batteries

Fig 4.2.1. Zigbee Module

4.2.4. ADVANTAGES OF ZIGBEE :

- Power saving, as a result of the short working period, low power consumption of communication and standby mode.
- Low cost of the modules and the zigbee protocol is patent fee free.
- Short time delay typically 30ms for device searching, 15ms for standby for activation.

4.2.5.DISADVANTAGES OF ZIGBEE :

- It can be used only for short range communication.
- It has slow data rate.
- High cost.

4.3. VOICE UNIT :

In voice unit IC APR9600 is used it is a voice recording and playback single chip IC. This IC is capable of recording for 60 seconds. It works of two modes auto rewind and record.

4.3.1. GENERAL DESCRIPTION:

APR9600 is a low-cost high performance sound record/replay IC incorporating flash analogue storage technique. Recorded sound is retained even after power supply is removed from the module. The replayed sound exhibits high quality with a low noise level. Sampling rate for a 60 second recording period is 4.2 kHz that gives a sound record/replay bandwidth of 20Hz to 2.1 kHz. However, by changing an oscillation resistor, a sampling rate as high as 8.0 kHz can be achieved. This shortens the total length of sound recording to 32 seconds.

Total sound recording time can be varied from 32 seconds to 60 seconds by changing the value of a single resistor. The IC can operate in one of two modes: serial mode and parallel mode.

In serial access mode, sound can be recorded in 256 sections. In parallel access mode, sound can be recorded in 2, 4 or 8 sections. The IC can be controlled simply using push button keys. It is also possible to control the IC using external digital circuitry such as micro-controllers and computers.

The APR9600 has a 28 pin DIP package. Supply voltage is between 4.5V to 6.5V. During recording and replaying, current consumption is 25 mA. In idle mode, the current drops to 1 µA.

The APR9600 experimental board is an assembled PCB board consisting of an APR9600 IC, an electret microphone, support components and

necessary switches to allow users to explore all functions of the APR9600 chip. The oscillation resistor is chosen so that the total recording period is 60 seconds with a sampling rate of 4.2 kHz. The board measures 80mm by 55mm.

Fig 4.3.1. APR9600 Experimental Board

4.3.2. APR9600 PIN EXPLANATION :

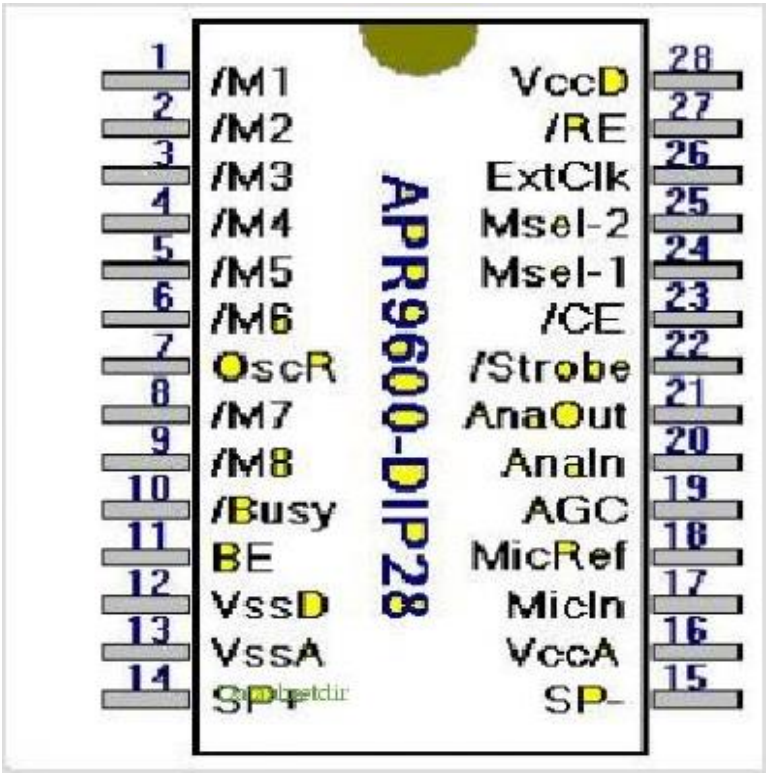


Fig 4.3.2. Pin Diagram APR9600

MSEL1	MSEL2	-M8	FUNCTION KEYS	FUNCTIONS
0	1	0 or 1	-M1, -M2 to select 1st and 2nd sound tracks. CE to stop	Parallel mode, 2 sections, 30 seconds for each
1	0	0 or 1	-M1 to –M4 to select a sound track, CE to stop	Parallel mode, 4 sections, 15 seconds for each

1	1	1	-M1 to –M8 to select a sound track, CE to stop	Parallel mode, 8 sections, 7.5 seconds for each
1	1	1	-M1 to –M8 to select a sound track, CE to stop	Pressing and hold down a key from –M1 to M8 to play the selected sound track repeatedly
0	0	1	-M1 and CE	Serial mode, allow up to 256 sound tracks to be recorded and played. Sound tracks are played from 1st to N in order after –M1 is toggled. Press CE to play from the 1st sound track
0	0	0	-M1,-M2 and CE	Serial mode, Press –M1 to replay one sound track. Toggle –M2 once to move to the next sound track. Press CE to play sound from the 1st sound track

Table 4.3.1 Modes & Selection Of Modes

PIN	**NAME**	**FUNCTION**	**PIN**	**NAME**	**FUNCTION**
1	-M1	Select 1st section of sound or serial mode recording and replaying control (low active)	15	SP-	Speaker negative end

2	-M2	Select 2nd section or fast forward control in serial mode (low active)	16	VCCA	Analogue circuit power supply
3	-M3	Select 3rd section of sound	17	MICIN	Microphone input (electret type microphone)
4	-M4	Select 4th section of sound	18	MICREF	Microphone reference input
5	-M5	Select 5th section of sound	19	AGC	AGC control
6	-M6	Select 6th section of sound	20	ANA IN	Audio input (accept a signal of 100 mV p-to-p)
7	OSCR	Resistor to set clock frequency.	21	ANA OUT	Audio output from the microphone amplifier
8	-M7	Select 7th section of sound or IC overflow indication	22	STROBE	During recording and replaying, it produces a strobe signal
9	-M8	Select 8th section of sound or select mode	23	CE	Reset sound track counter to zero/ Stop or Start / Stop
10	BUSY	Busy (low active)	24	MSEL1	Mode selection 1

11	BE	=1, beep when a key is pressed =0, do not beep	25	MSEL2	Mode selection 2
12	VSSD	Digital circuit ground	26	EXTCLK	External clock input
13	VSSA	Analog circuit ground	27	-RE	=0 to record, =1 to replay
14	SP+	Speaker, positive end	28	VCCD	Digital circuit power supply.

Table 4.3.2 Pin Description Of Apr9600

Notes:

• RE=0 to record sound. RE=1 to replay sound

• Press -M1 to -M8 once to replay a sound track. Press the key again to stop replaying the track

• Press and hold –M1 to -M8 continuously, the corresponding track will be replayed repeatedly

• During recording, -M1 to M8 should be pressed while the sound is being recorded. Releasing the key terminates recording.

4.3.3. FEATURES OF APR9600 :

- Single chip, high quality voice recording &play back solutions.
  - No external ICs required.
  - Minimum external components
- Non-volatile flash memory technology
  - No battery backup required.
- User selectable messaging options.

- Random access of multiple fixed duration messages
- Sequential access of multiple variable duration messages.

- User-friendly, easy to use operation .
  - Programming and developing systems not required.
  - Level activated recording & edge activated playback switches.
- Low power consumption
  - Operating current: 25mA typical.
  - Standby current: 1uA typical.
  - Automatic power down.
- Chip enable pin for simple message expansion.

CHAPTER 5 : SOFTWARE EXPLANATION (ENERGIA)&CODING

5.1. INTRODUCTION:

Energia is an open-source electronics prototyping platform started by Robert Wessels in January of 2012 with the goal to bring the Wiring and Arduino framework to the Texas Instruments MSP430 based LaunchPad. The Energia IDE is cross platform and supported onMac OS, Windows, and Linux. Energia uses the mspgcc compiler by Peter Bigot and is based on the Wiring and Arduino framework. Energia includes an integrated development environment (IDE) that is based on Processing.

The foundation of Energia and Arduino is the Wiring framework that is developed by Hernando Barragan. The framework is thoughtfully created with designers and artists in mind to encourage a community where both beginners and experts from around the world share ideas, knowledge and their collective experience. The Energia team adopts the philosophy of learning by doing and strives to make it easy to work directly with the hardware. Professional engineers,entrepreneurs, makers, and students can all benefit from the ease of use Energia brings to the microcontroller.

Energia started out to bring the Wiring and Arduino framework to the Texas Instruments MSP430 LaunchPad. Texas Instruments offers a MSP430, TM4C, C2000, andCC3200 LaunchPad. The LaunchPad is a low-cost microcontroller board that is made by Texas Instruments. The latest release of Energia supports the majority of the LaunchPad product offerings.

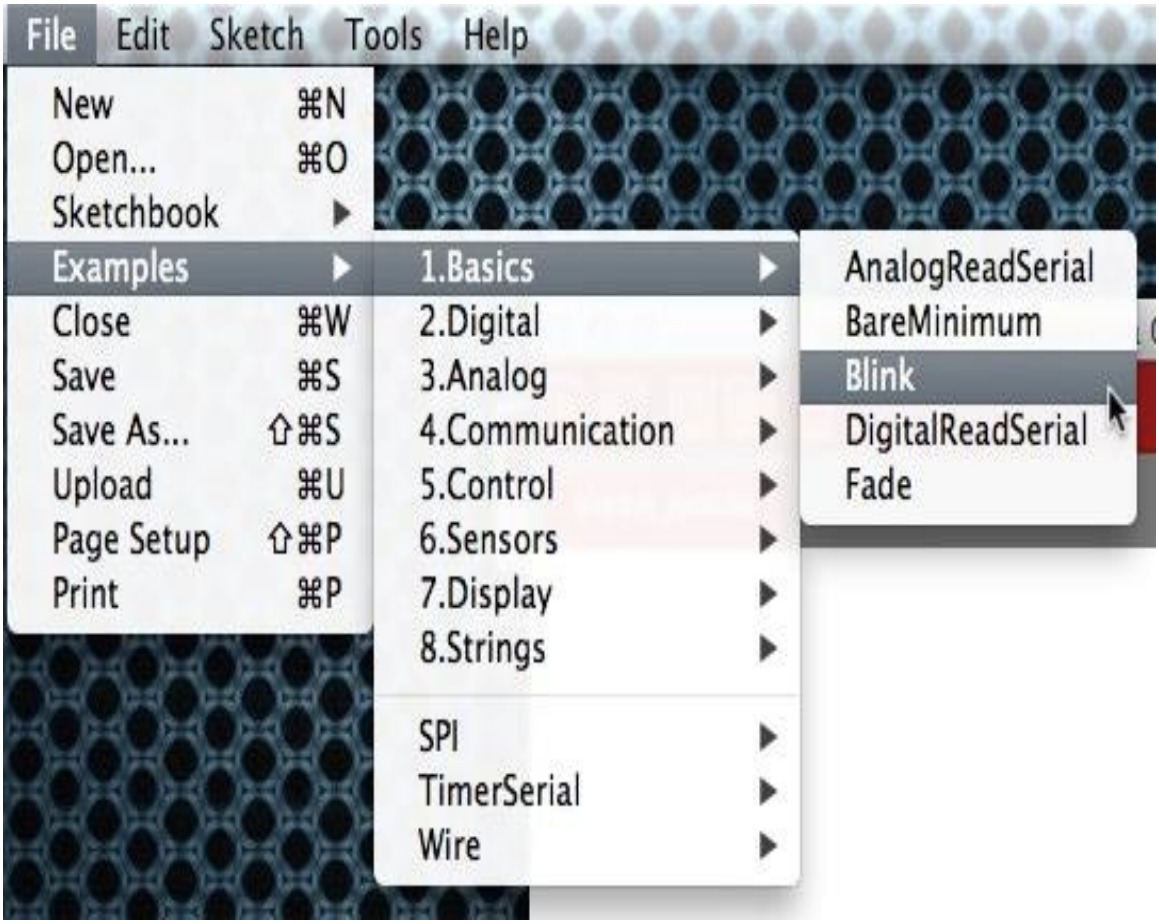


Fig No:5.1. Energia Software

5.2. CORE FUNCTIONS:

Simple programs that demonstrate basic Energia commands. These are included with the Energia environment; to open them, click the Open button on the toolbar and look in the examples folder.

5.2.1. BASICS:

- BareMinimum: The bare minimum of code needed to start an Energia sketch.
- Blink: Turn an LED on and off.
- DigitalReadSerial: Read a switch, print the state out to the Energia Serial Monitor.

- AnalogReadSerial: Read a potentiometer, print it's state out to the Energia Serial Monitor.
- Fade: Demonstrates the use of analog output to fade an LED.
- ReadAnalogVoltage : Reads an analog input and prints the voltage to the serial monitor

5.2.2. DIGITAL:

- Blink Without Delay: blinking an LED without using the delay() function.
- Button: use a pushbutton to control an LED.
- Debounce: read a pushbutton, filtering noise.
- Button State Change: counting the number of button pushes.
- Input Pullup Serial: Demonstrates the use of INPUT_PULLUP with pinMode().
- Tone: play a melody with a Piezo speaker.
- Pitch follower: play a pitch on a piezo speaker depending on an analog input.
- Simple Keyboard: a three-key musical keyboard using force sensors and a piezo speaker.
- Tone4: play tones on multiple speakers sequentially using the tone() command.

5.2.3. ANALOG:

- AnalogInOutSerial: read an analog input pin, map the result, and then use that data to dim or brighten an LED.
- Analog Input: use a potentiometer to control the blinking of an LED.
- AnalogWrite: fade 7 LEDs on and off, one by one, using an MSP430G2 LaunchPad board.
- Calibration: define a maximum and minimum for expected analog sensor values.
- Fading: use an analog output (PWM pin) to fade an LED.
- Smoothing: smooth multiple readings of an analog input.

5.2.4.COMMUNICATION:

These examples include code that allows the LaunchPad to talk to Processing sketches running on the computer.

- ReadASCIIString: parse a comma-separated string of ints to fade an LED
- ASCII Table: demonstrates Energia's advanced serial output functions.
- Dimmer: move the mouse to change the brightness of an LED.
- Graph: send data to the computer and graph it in Processing.
- Physical Pixel: turn a LED on and off by sending data to your LaunchPad from Processing.
- Virtual Color Mixer: send multiple variables from LaunchPad to your computer and read them in Processing.
- Serial Call Response: send multiple vairables using a call-and-response (handshaking) method.
- Serial Call Response ASCII: send multiple variables using a call-and-response (handshaking) method, and ASCII-encode the values before sending.
- SerialEvent: Demonstrates the use of SerialEvent().
- Serial input (Switch (case) Statement): how to take different actions based on characters received by the serial port.

5.2. 5.CONTROL STRUCTURES:

If Statement (Conditional): how to use an if statement to change output conditions based on changing input conditions.

- For Loop: controlling multiple LEDs with a for loop.
- Array: a variation on the For Loop example that demonstrates how to use an array.
- While Loop: how to use a while loop to calibrate a sensor while a button is being read.

- Switch Case: how to choose between a discrete number of values. Equivalent to multiple If statements. This example shows how to divide a sensor's range into a set of four bands and to take four different actions depending on which band the result is in.
- Switch Case 2: a second switch-case example, showing how to take different actions based in characters received in the serial port.

5.2.6. STRINGS:

- StringAdditionOperator: add strings together in a variety of ways.
- StringAppendOperator: append data to strings.
- StringCaseChanges: change the case of a string.
- StringCharacters: get/set the value of a specific character in a string.
- StringComparisonOperators: compare strings alphabetically.
- StringConstructors: how to initialize string objects.
- StringIndexOf: look for the first/last instance of a character in a string.
- StringLength & StringLengthTrim: get and trim the length of a string.
- StringReplace: replace individual characters in a string.
- StringStartsWithEndsWith: check which characters/substrings a given string starts or ends with.
- StringSubstring: look for "phrases" within a given string.

5.2.7. SENSORS, MOTORS & DISPLAYS:

Temperature: use on board MCU core temp sensor.

- Tilt Sensor: use a basic tilt sensor.
- Servo: move a servo to control mechanical objects.
- Basic Motor: turn a basic motor.

- 7 Segment Display: display basic number and letter values.
- 2×16 Character Display: output strings to a character display.

5.2.8. BOOSTERPACKS:

- Olimex8x8matrix: create a scrolling marquee with an LED matrix.
- Sharp LCD Display: display images and text on the low power LCD
- EducationalBP: create a magic 8 ball with an accelerometer and LCD display.
- EducationalBP MKII: examples involving buzzer, LCD, LEDs, accelerometer, push buttons, and more
- CC3000: introduction to SimpleLink WiFi CC3000 BoosterPack
- CC3100: introduction to SimpleLink WiFi CC3100 BoosterPack

5.2.9. CONNECTIVITY:

- WiFi: WiFi library examples
- MQTT: Use the MQTT lightweight protocol to enable IoT & M2M applications
- StandardFirmata: Use firmata protocol to dynamically communicate with the microcontroller
- Temboo: Access hundreds of web APIs through Temboo using Energia
- AT&T M2X: Post Energia data to AT&T M2X cloud service
- BLE Mini: Use Red Bear Lab BLE Mini to control you LaunchPad
- Freeboard.io: Create a cloud dashboard with your Energia data using freeboard.io
- Contiki: Access Contiki OS for IoT using Energia

5.3 CODING

```
unsigned long int fs1=A1;
unsigned long int fs2=A2;
unsigned long int fs3=A3;
unsigned long int mx=A4;

unsigned long int fs1read=0;
unsigned long int fs2read=0;
unsigned long int fs3read=0;
unsigned long int mxread=0;


int sensorValue1 = 0;        // value read from the pot
int sensorValue2 = 0;        // value read from the pot
int sensorValue3 = 0;        // value read from the pot
int sensorValue4 = 0;        // value read from the pot

int outputValue1 = 0;        // value output to the PWM (analog out)
int outputValue2 = 0;        // value output to the PWM (analog out)
int outputValue3 = 0;        // value output to the PWM (analog out)
int outputValue4 = 0;        // value output to the PWM (analog out)
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
```

```
void loop() {
  // read the analog in value:
  fs1read = analogRead(fs1)/10;
  fs2read=analogRead(fs2)/10;
  fs3read=analogRead(fs3)/10;
  mxread=analogRead(mx);
  // map it to the range of the analog out:
  outputValue1 = map(sensorValue1, 0, 1023, 0, 255);
  outputValue2 = map(sensorValue2, 0, 1023, 0, 255);
  outputValue3 = map(sensorValue3, 0, 1023, 0, 255);
  outputValue4 = map(sensorValue4, 0, 1023, 0, 255);
  // change the analog out value:

if(fs1read<=125 && fs2read)
  {
  Serial.print("fs1 = " );
  Serial.print(fs1read);
Serial.print(“1”);
 delay(20);
  }
  else if(fs2read<=125)
  {
    Serial.print("fs2 = ");
    Serial.print(fs2read);
    Serial.print(“2”);
    delay(20);
```

```
  }
   else if(fs3read<90)
  {
    Serial.print("fs3=");
    Serial.print(fs3read);
   Serial.print(“3”);
    delay(20);
  }

  else if(mxread>250)
   {
    Serial.print("mems =");
     Serial.print(mxread);
   Serial.print(“4”);
   delay(20);
   }

}

int play,v1,v2,v3,v4,v5;
int inByte = 0; // incoming serial byte

void setup()
{
  // start serial port at 9600 bps:
  Serial.begin(9600);
```

```
  pinMode(11, INPUT); // digital sensor is on digital pin 11
  establishContact(); // send a byte to establish contact until receiver responds
}

void loop()
{
  play=v1=v2=v3=v4=v5=0;
  // if we get a valid byte, read analog ins:
  if (Serial.available())
  {
    // get incoming byte:
    inByte = Serial.read();

  }  // read first analog input, divide by 4 to make the range 0-255:
  if(inByte==1)
 {
  v1=0;delay(1000);play=0;delay(2000);play=1;v1=1;
 }
else if(inByte==2)
{
  v2=0;delay(1000);play=0;delay(2000);play=1;v2=1;
}
else if(inByte==3)
{
  v3=0;delay(1000);play=0;delay(2000);play=1;v3=1;
}
```

```
else if(inByte==4)
{
  v4=0;delay(1000);play=0;delay(2000);play=1;v4=1;
}
else if(inByte==5)
{
  v5=0;delay(1000);play=0;delay(2000);play=1;v5=1;
}


}
  void establishContact() {
  while (Serial.available() <= 0) {
  Serial.println('A'); // send a capital A
  delay(300);
  }
}
```

CHAPTER 6 : RESULTS& CONCLUSION

6.1 DETECTION UNIT :

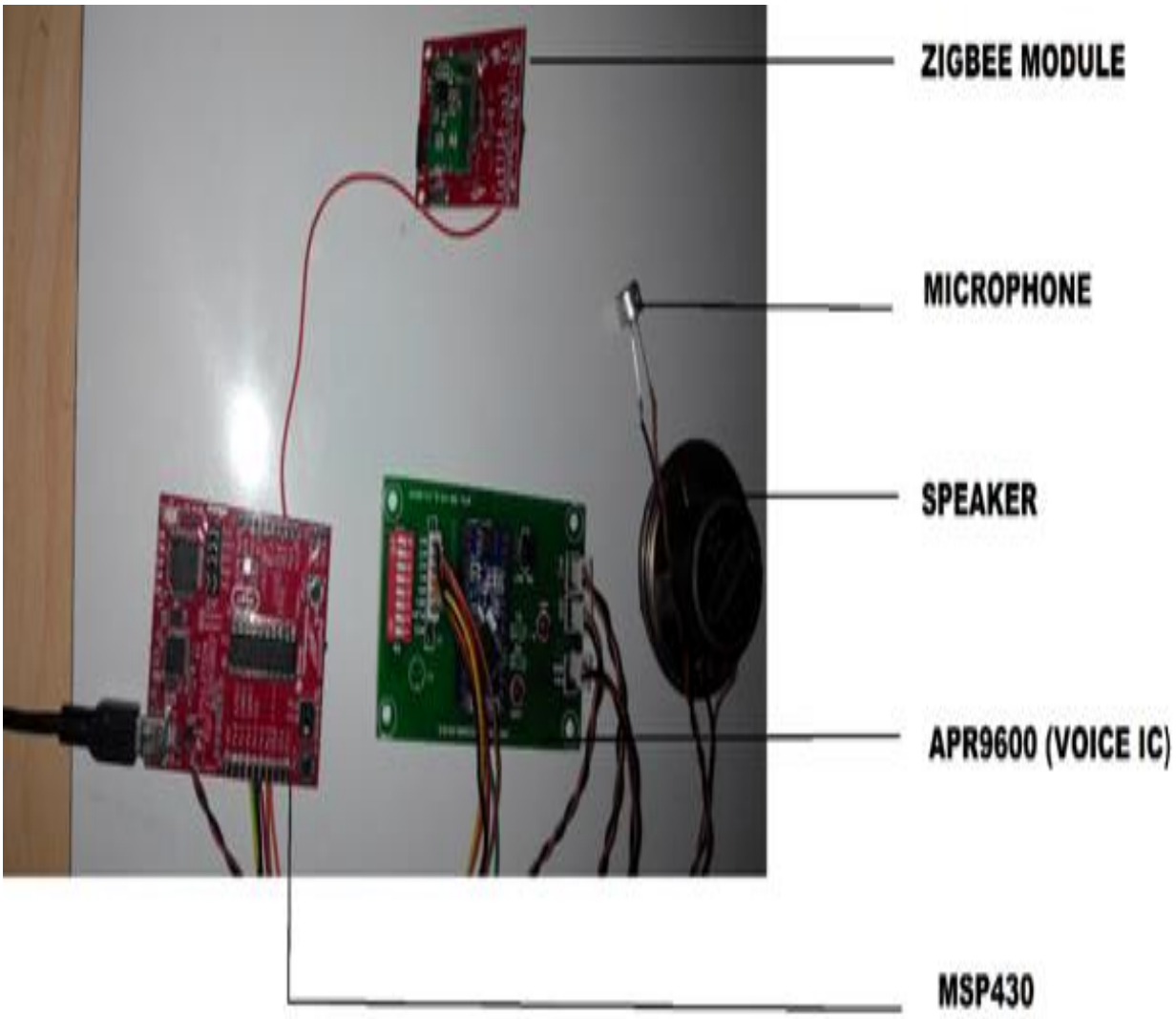


Fig 6.1 Detection Unit Hardware Set Up

As described in our previous sections the detection unit is connected.

6.2 VOICE UNIT :

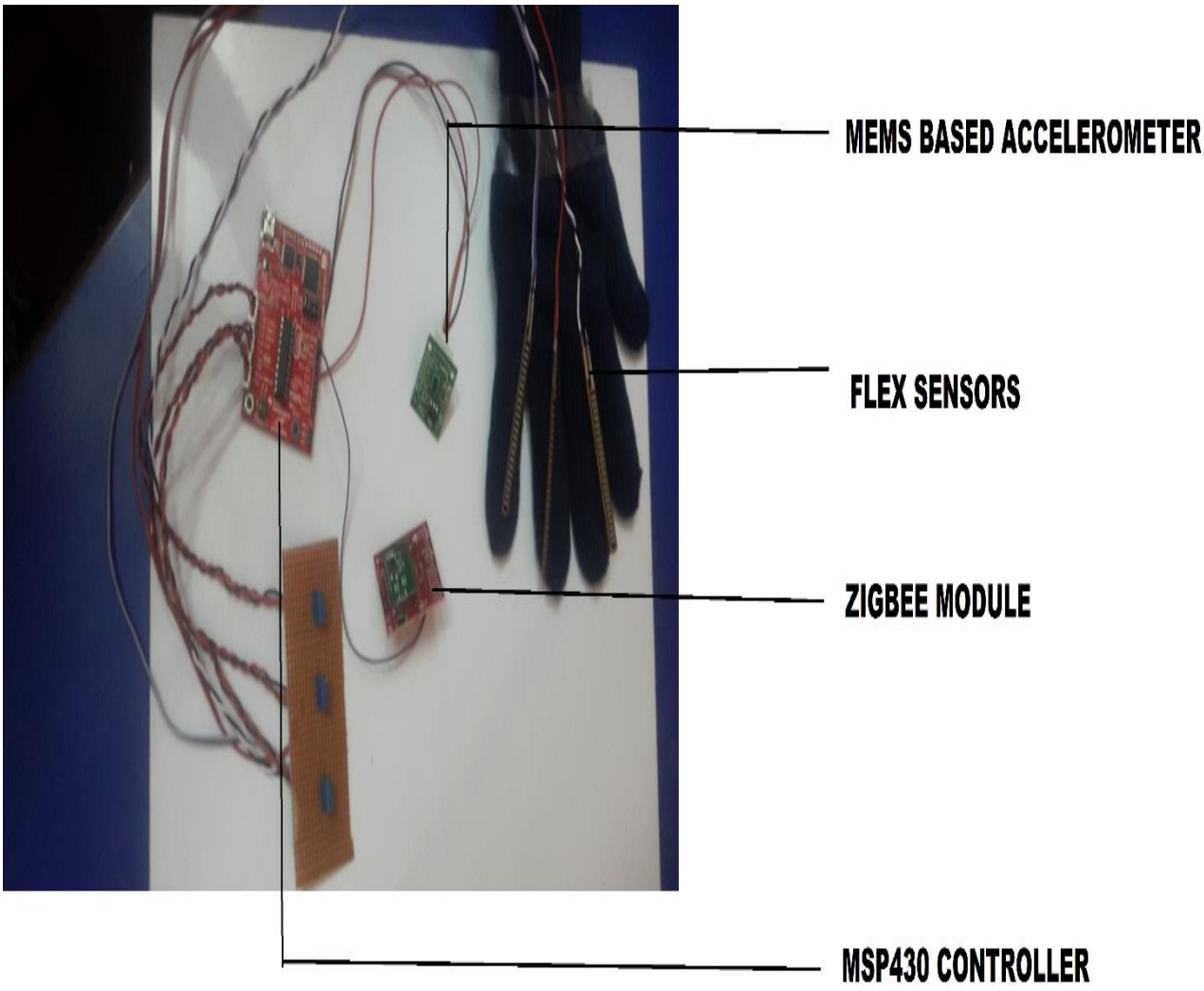


Fig6.2 Voice Unit Hardware Set Up

As the flex sensor attached to the hand glove are moved the voice recorded using microphone is played using speaker.

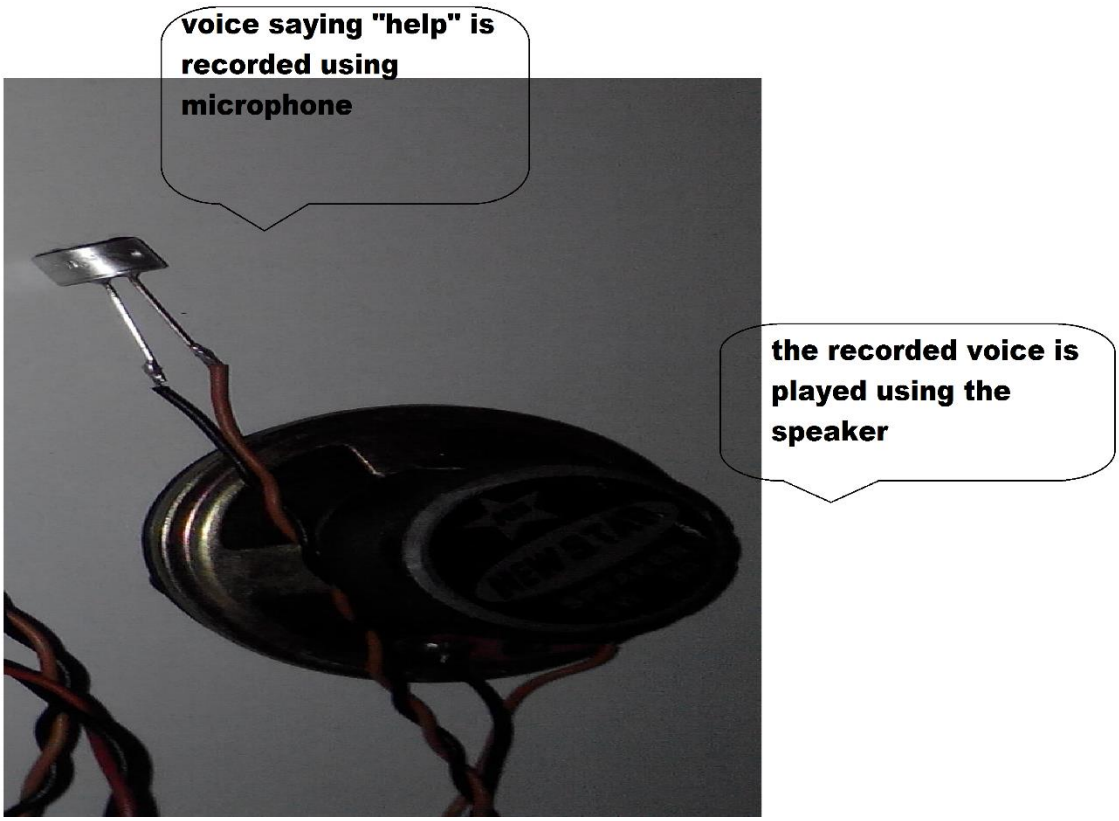


Fig 6.3 Working Description.

When a particular hand gesture is made the voice saying “help” or any other word of one’s choice are recorded using the microphone in the record mode of APR9600.

After recording the voice unit is switched to play mode. In this mode when the same hand gesture is made the voice recorded corresponding to the gesture is played using speaker.

This system has the convenience of recording different languages of one’s choice. Since this is a prototype 6 to 8 voices are recorded.

In our proposed system, Flex sensors and MEMS are used for accurate hand gesture recognition. The analog signals from these sensors are converted into digital using msp430 and they are transmitted using zigbee to the voice unit apr9600.Further enhancements that can be done in this system are, by using additional sensors like contact sensors the combinations can be increased and many sign can be stored. Zigbee is short range communication device this system is made efficient by enabling long range communications.

REFERENCES

- S. Mitra and T. Acharya "Gesture recognition: A survey", *IEEE Trans. Syst., Man,Cybern., Part C, Appl. Rev.*, vol. 37, no. 3, pp.311 -324 2007
- M. A. Amin and H. Yan "Sign language finger alphabet recognition from gabor-PCA representation of hand gestures", *Proc. 6th Int. Conf. Mach. Learn.Cybern.*, pp.2218 -2223 2007
- S. Zhou "Gesture recognition for interactive controllers using MEMS motion sensors", *Proc. 4th IEEE Int. Conf. Nano/Micro Eng. Molecular Syst.*, pp.935 -940 2009
- M. Elmezain, A. Al-Hamadi , S. S. Pathan and B. Michaelis "Spatio-temporal feature extraction-based hand gesture recognition for isolated American sign language and Arabic numbers", *Proc. 6th Int. Symp. Image Signal Process. Anal.*, pp.254 -259 2009
- M. G. Ceruti "Wireless communication glove apparatus for motion tracking, gesture recognition, data transmission, and reception in extreme environments", Proc. *ACMSymp. Appl. Comput.*, pp.172 -176 2009
- J. S. Wang and F. C. Chuang "An accelerometer-based digital pen with a trajectory recognition algorithm forhandwritten digit and gesture recognition", *IEEE Trans. Ind. Electron.*, vol. 59, no. 7, pp.2998 -3007 2012
- R. Xu , S. Zhou and W. J. Li "MEMS accelerometer based nonspecific-user hand gesture recognition", *IEEE Sensors J.*, vol. 12, pp.1166 -1173 2012
- A. Seniuk and D. Blostein "Pen acoustic emissions for text and gesturerecognition", *Proc. 10th Int. Conf. Document Anal. Recognit.*, pp.872 -876 2009
- P. Asadzadeh , L. Kulik and E. Tanin "Gesture recognition using RFID technology", *Pers. UbiquitousComput.*, vol. 16, no. 3, pp.225 -234 2012

- C. Sung-do and L. Soo-Young "3D stroke reconstruction and cursivescript recognition with magnetometer-aided inertial measurement unit", *IEEE Trans.Consumer Electron.*, vol. 58, pp.661 -669 2012
- W. De Vries , H. Veeger , C. Baten and F. Van De Helm "Magnetic distortion in motion labs, implications for validating inertialmagnetic sensors", *Gait Posture*, vol. 29, no. 4, pp.535 -541 2009
- N. Phuong , H. J. Kang , Y. S. Suh and Y. S. Ro "A DCM based orientation estimation algorithm with an inertial measurement unit and a magnetic compass", *J. Univ. Comput.Sci.*, vol. 15, no. 4, pp.859 -876 2009
- H. M. Schepers , D. Roetenberg and P. H. Veltink "Ambulatory human motion tracking by fusion of inertial and magneticsensing with adaptive actuation", *Med. Biol. Eng. Comput.*, vol. 48, no. 1, pp.27 -37 2010
- W. T. Faulkner , R. Alwood , D. W. A. Taylor and J. Bohlin "GPS-denied pedestrian tracking in indoor environments using an IMU and magnetic compass", *Proc. Int. Tech. Meeting Inst. Navigat.*, pp.198 -204 2010
- A. Jimenez, F. Seco, C. Prieto and J. Guevara "A comparison of pedestriandead-reckoning algorithms using a low-cost MEMS IMU", *Proc. IEEE Int. Symp. Intell. SignalProcess.*, pp.37 -42 2009
- D. Dusha and L. Mejias "Error analysis and attitudeobservability of a monocular GPS/visual odometry integrated navigation filter", *Int. J.Robot. Res.*, vol. 31, no. 6, pp.714 -737 2012
- J. Naranjo , F. Jimnez , F. Aparicio and J. Zato "GPS and inertial systems for highprecision positioning on motorways", *J. Navigat.*, vol. 62, no. 2, pp.351 -363 2009
- C.H. Kang , S. Y. Kim and C. G. Park "Improvement of a low cost mems inertial-GPS integrated system using wavelet denoisingtechniques", *Int. J. Aeronautical Space Sci.*, vol. 12, no. 4, pp.371 -378 2011

- R.E. Hopkins , N. M. Barbour , D. E. Gustafson and P. Sherman *Miniature Inertial and Augmentation Sensorsfor Integrated Inertial/GPS Based Navigation Applications*, 2010
- J. Hol *Sensor fusion and calibration of inertial sensors, vision, ultra wideband and GPS*, 2011
- G.Bleser and D. Stricker "Advanced tracking through efficient image processing and visual-inertial sensor fusion", *Comput. Graph.*, vol. 33, no. 1, pp.59 -72 2009
- G. Bleser and G. Hendeby "Using optical flow for filling the gaps in visual-inertial tracking", *Proc. EUSIPCO*, pp.1057 -1063 2010
- R. Z. Khan and N. A. Ibraheem "Comparative study of hand gesture recognition system", *Proc. Comput.Sci. Inf. Technol.*, pp.203 -213 2012
- Y. Tao, H. Hu and H. Zhou "Integration of vision and inertial sensors for 3D arm motion tracking in home-based rehabilitation", *Int. J. Robot. Res.*, vol. 26, no. 6, pp.607 -624 2007
- Y. Suya , U. Neumann and R. Azuma "Hybrid inertial and vision tracking for augmented reality registration", *Proc. Virtual Reality*, pp.260 -267 1999
- S. Giannarou, Z. Zhiqiang and Y. Guang-Zhong "Deformable structure from motion by fusing visual and inertial measurement data", *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp.4816 -4821 2012
- P. Gemeiner, P. Einramhof and M. Vincze "Simultaneous motion and structure estimation by fusion of inertial and vision data", *Int. J. Robot. Res.*, vol. 26, no. 6, pp.591 -605 2007
- L. Armesto, J. Tornero and M. Vincze "Fast ego-motion estimation with multi-rate fusion of inertial and vision", *Int. J. Robot. Res.*, vol. 26, no. 6, pp.577 -589 2007