1.Introduction

1.1 Purpose:

The purpose of this project is to design a smart blind stick which can provide guidelines to the blind/deaf and dumb peoples and also provides security from different situations like obstacle, manhole and water.

Voice circuit is a system which is capable of storing voices and playing back the stored voices when requested. This system eliminates the usage of old alarm systems and makes to configure the alerts through voice.

This consists of a Microcontroller based control system, Voice Module, water sensor,manwhole detection sensor and Obstacle detection Sensor. This device senses the obstacles in its path by continuously transmitting the IR rays from IR transmitter. If any obstacle comes in its vicinity then the rays are destructed and give this input to the microcontroller. The IR receiver fitted on the device senses these Obstacles and this information is passed onto the Microcontroller. If any manhole or water detect in front of the person then this system immediately gives alert to the person.

1.2 Problem definition:

Helping Hands for the Blind was founded to address the concerns, and as a vital resource for the blind. Helping Hands for the Blind is an organization of blind people who want to help other blind people. It is a problem solving organization. It is a guide that blind people can turn to in times of need. Helping a blind person through all the paperwork can literally be a life saver. Another important form of assistance is to provide a Mobility Instructor. When a blind person is new to an area, it is important that they be shown how to get around by a trained and knowledgeable instructor. There is a large and growing demand for this service. The project aims in designing a system which is capable of alerting the user if his destination is reached.

1.3 Existing system:

In Existing We are Designing Stick Integrared with Bell. The Person has to pass the instructions with that bell if any obstacle will come means they are not gng to identify and we have to face the issues to resolves that issues we are implementing our project.

Disadvantages of existing system:

- In existing system obstrical detection and alerting the people is not present.
- There is no Tracking of Water and Manhole while walking.

1.4 Proposed system:

In proposed system this blind stick have the capability to detect any kind of obstacles using ultrasonic sensors, and also can detect manholes using IR sensor,Wtaer in front of the blind. This blind stick can monitor and alert the blind through vibration and buzzer alarm from multiple situations.and updates it to iot.

Advantages of proposed system:

- The system can be used both indoor and outdoor navigation.
- Blind person's can be tracked whenever needed which will ensure additional safety.
- Detects obstacles and alerts the blind person through vibration alert and speech output.

2.Requirement analysis

Requirement analysis is done in order to understand the problem of the software system is to solve. The problem could be automating an existing manual process, developing a new automated system, or a combination of two. For large systems that have many features, and that need to perform many different tasks, understanding the requirements of the system is a major task.

2.1 FUNCTIONAL REQUIREMENTS:

The functional requirements describe the interaction between the system and its environment, independent of its implementation and services provided by the system.

2.2 SOFTWARE REQUIREMENTS:

1. PIC-C compiler for Embedded C programming.
2. PIC kit 2 programmer for dumping code into Micro controller.
3. Express SCH for Circuit design.
4. Proteus for hardware simulation.
5. Arm7 IDE Compiler.
6. Atmel programmer for dumping code into Micro controller.
7. Mobile /Desktop Application.

Embedded System

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. A good example is the microwave oven. Almost every household has one, and tens of millions of them are used everyday, but very few people realize that a processor and software are involved in the preparation of their lunch or dinner.

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things.

Frequently, an embedded system is a component within some larger system. For example, modern cars and trucks contain many embedded systems. One embedded system controls the anti-lock brakes, other monitors and controls the vehicle's emissions, and a third displays information on the dashboard. In some cases, these embedded systems are connected by some sort of a communication network, but that is certainly not a requirement.

For example, my computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive, and sound card-each of which is an embedded system. Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over analog telephone line. That's it and all of the other devices can be summarized in a single sentence as well.

If an embedded system is designed well, the existence of the processor and software could be completely unnoticed by the user of the device. Such is the case for a microwave oven, VCR, or alarm clock. In some cases, it would even be possible to build an equivalent device that does not contain the processor and software. This could be done by replacing the combination with a custom integrated circuit that performs the same functions in hardware. However, a lot of flexibility is lost when a design is hard-cooled in this way. It is mush easier, and cheaper, to change a few lines of software than to redesign a piece of custom hardware

2.3 Hardware Requirements

2.3.1 ARM based LPC2148

The LPC2148 microcontrollers are based on a 32/16 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at the maximum clock rate. For critical code size applications, the alternative 16-bit Thumb mode reduces code by more than 30 % with minimal performance penalty.

Due to their tiny size and low power consumption, LPC2141/2/4/6/8 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. A blend of serial communications interfaces ranging from a USB 2.0 Full Speed device, multiple UARTS, SPI, SSP to I2Cs and on-chip SRAM of 8 kB up to 40 kB, make these devices very well suited for communication gateways and protocol converters, soft modems, voice recognition and low end imaging, providing both large buffer size and high processing power. Various 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins make these microcontrollers particularly suitable for industrial control and medical systems.

2.3.2 Features

• 16/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.

• 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory.128 bitwide interface/accelerator enables high speed 60 MHz operation.

• In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1 ms.

• Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high speed tracing of instruction execution.

• USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provides 8 kB of on-chip RAM accessible to USB by DMA.

• One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14 analog inputs, with conversion times as low as 2.44 μs per channel.

• Single 10-bit D/A converter provide variable analog output

2.3.3Applications

• Industrial control

• Medical systems

• Access control

• Point-of-sale

• Communication gateway

• Embedded soft modem

2.3.4Device information

Device	Number of pins	On-chip SRAM	Endpoint USB RAM	On-chip FLASH	Number of 10-bit ADC channels	Number of 10-bit DAC channels	Note
LPC2141	64	8 kB	2 kB	32 kB	6	-	-
LPC2142	64	16 kB	2 kB	64 kB	6	1	-
LPC2144	64	16 kB	2 kB	128 kB	14	1	UART1 with full modem interface
LPC2146	64	32 kB + 8 kB[1]	2 kB	256 kB	14	1	UART1 with full modem interface
LPC2148	64	32 kB + 8 kB[1]	2 kB	512 kB	14	1	UART1 with full modem interface

2.4 Architectural overview

The LPC2148 consists of an ARM7TDMI-S CPU with emulation support, the ARM7 Local Bus for interface to on-chip memory controllers, the AMBA Advanced High-performance Bus (AHB) for interface to the interrupt controller, and the VLSI Peripheral Bus (VPB, a compatible superset of ARM's AMBA Advanced Peripheral Bus) for connection to on-chip peripheral functions. The LPC2148 configures the ARM7TDMI-S processor in little-endian byte order.

AHB peripherals are allocated a 2 megabyte range of addresses at the very top of the 4 gigabyte ARM memory space. Each AHB peripheral is allocated a 16 kB address space within the AHB address space. LPC2148 peripheral functions (other than the interrupt controller) are connected to the VPB bus. The AHB to VPB bridge interfaces the VPB bus to the AHB bus. VPB peripherals are also allocated a 2 megabyte range of addresses, beginning at the 3.5 gigabyte address point.

2.4.1 ARM7TDMI-S Processor

The ARM7TDMI-S is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and

related decode mechanism are much simpler than those of micro programmed Complex Instruction Set Computers. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor core.

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

On-chip Flash memory system

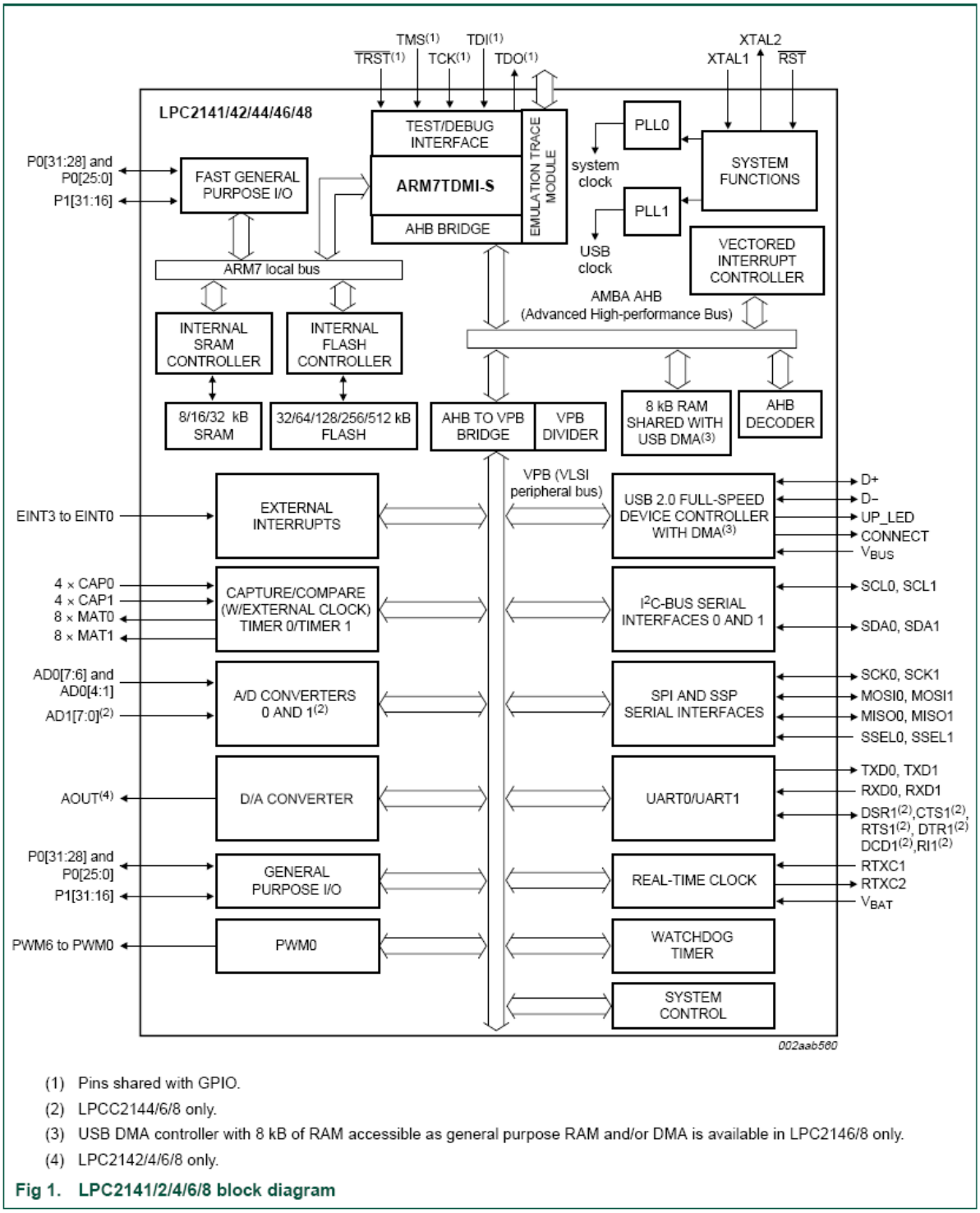
The LPC2141/2/4/6/8 incorporates a 32 kB, 64 kB, 128 kB, 256 kB, and 512 kB Flash memory system respectively. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways: over the serial built-in JTAG interface, using In System Programming (ISP) and UART0, or by means of In Application Programming (IAP) capabilities.

The application program, using the IAP functions, may also erase and/or program the Flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc. When the LPC2148 on-chip boot loader is used, 32 kB, 64 kB, 128 kB, 256 kB, and 500 kB of Flash memory is available for user code. The LPC2148 Flash memory provides minimum of 100,000 erase/write cycles and 20 years of data-retention.

On-chip Static RAM (SRAM)

On-chip Static RAM (SRAM) may be used for code and/or data storage. The on-chip SRAM may be accessed as 8-bits, 16-bits, and 32-bits. The LPC2148 provide 8/16/32 kB of static RAM respectively. The LPC2148 SRAM is designed to be accessed as a byte-addressed memory.

2.4.2 Block diagram



(1) Pins shared with GPIO.
(2) LPCC2144/6/8 only.
(3) USB DMA controller with 8 kB of RAM accessible as general purpose RAM and/or DMA is available in LPC2146/8 only.
(4) LPC2142/4/6/8 only.

Fig 1. LPC2141/2/4/6/8 block diagram

Fig No :1.1 LPC2148 Block Diagram

2.4.3 Memory re-mapping

In order to allow for compatibility with future derivatives, the entire Boot Block is mapped to the top of the on-chip memory space. In this manner, the use of larger or smaller flash modules will not require changing the location of the Boot Block (which would require changing the Boot Loader code itself) or changing the mapping of the Boot Block interrupt vectors. Memory spaces other than the interrupt vectors remain in fixed locations. Shows the on-chip memory mapping in the modes defined above.

The portion of memory that is re-mapped to allow interrupt processing in different modes includes the interrupt vector area (32 bytes) and an additional 32 bytes, for a total of 64 bytes. The re-mapped code locations overlay addresses 0x0000 0000 through 0x0000 003F. A typical user program in the Flash memory can place the entire FIQ handler at address 0x0000 001C without any need to consider memory boundaries. The vector contained in the SRAM, external memory, and Boot Block must contain branches to the actual interrupt handlers, or to other instructions that accomplish the branch to the interrupt handlers. There are three reasons this configuration was chosen:

1. To give the FIQ handler in the Flash memory the advantage of not having to take a memory boundary caused by the remapping into account.
2. Minimize the need to for the SRAM and Boot Block vectors to deal with arbitrary boundaries in the middle of code space.
3. To provide space to store constants for jumping beyond the range of single word branch instructions.

Re-mapped memory areas, including the Boot Block and interrupt vectors, continue to appear in their original location in addition to the re-mapped address.

4. Minimize the need to for the SRAM and Boot Block vectors to deal with arbitrary boundaries in the middle of code space.
5. To provide space to store constants for jumping beyond the range of single word branch instructions.Re-mapped memory areas, including the Boot Block and interrupt vectors, continue to appear in their original location in addition to the re-mapped address.

2.1.7 Pin description for LPC2148

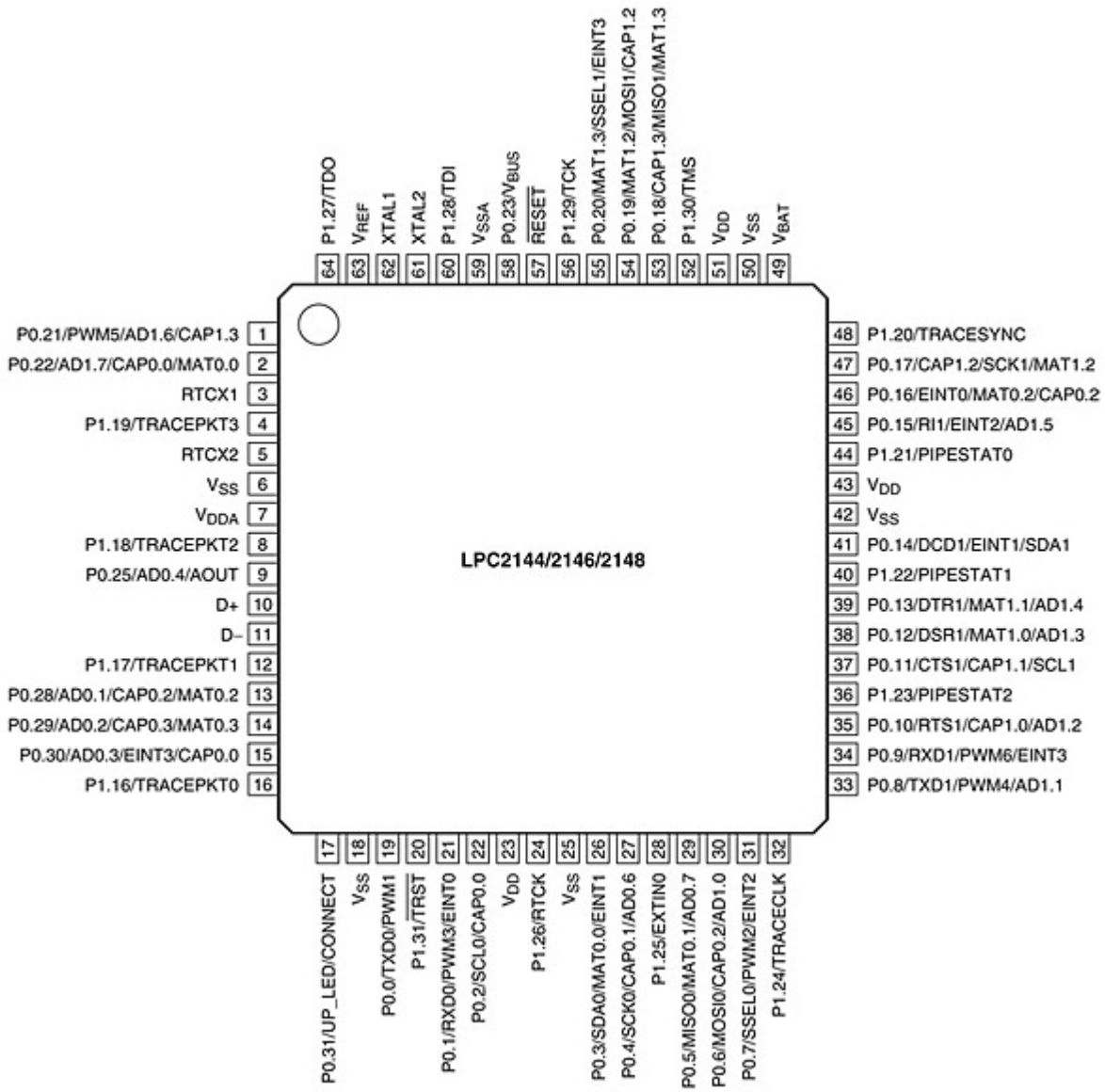


Fig no:1.3 Pin Description

Pin Description

TheLPC2148 processor has totally four ports.

1. Port0 has 32 pins and all can be used as input/output. All pins of this port can be used as general purpose input/output. The number of pins available for input/output operation will depends on the use of alternate functions i.e. if we

use less alternate functions more are the available input/output's **[1]**.Port Pins P0.24,P0.26,P0.27 are not available.

2. Port1 has16 pins and all can be used as input/output. All pins of this port can be used as general purpose input/output. This is same as port0, only difference is this port has only 16pins where as port0 has 32 pins.

Pin Name	Type	Description
P0.0 – P0.31 P1.16 – P1.31	Input/ Output	General purpose input/output. The number of GPIOs available depends on the use of functions.

Table 1: Pin Description

2.5 POWER SUPPLY

All digital circuits require regulated power supply. In this article we are going to learn how to get a regulated positive supply from the mains supply.

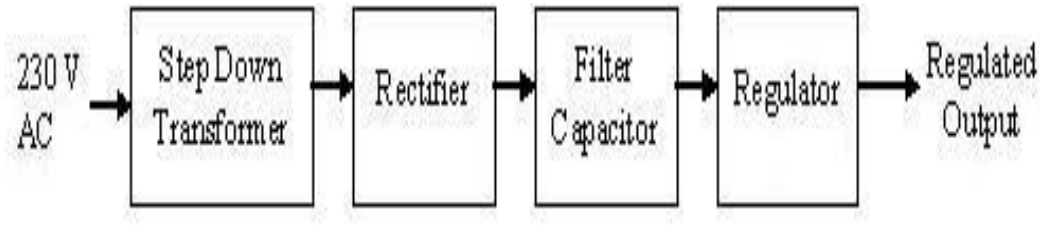


Fig no: 1.4 Power Supply

It shows the basic block diagram of a fixed regulated power supply. Let us go through each block.

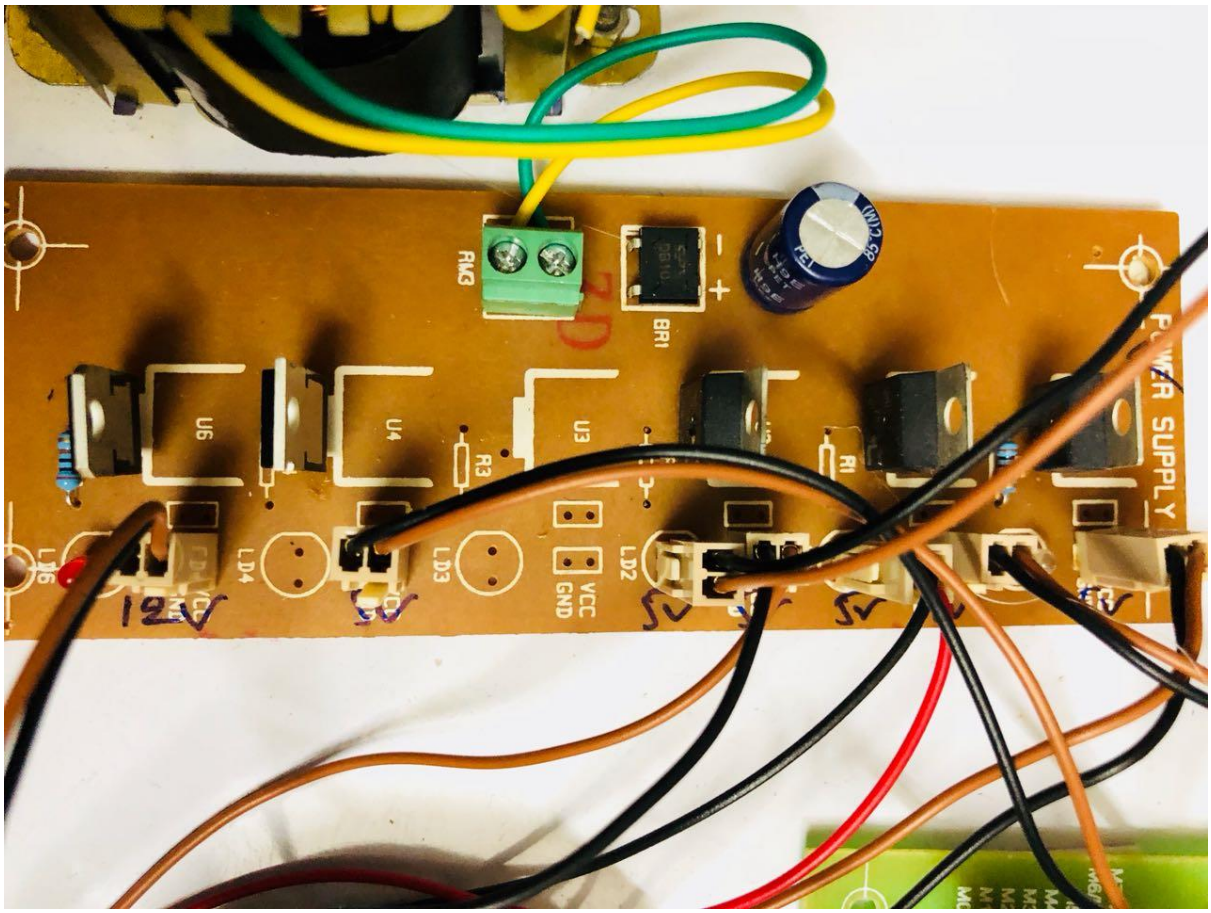


Fig No : 1.5 Power Supply Board

2.6 TRANSFOMER

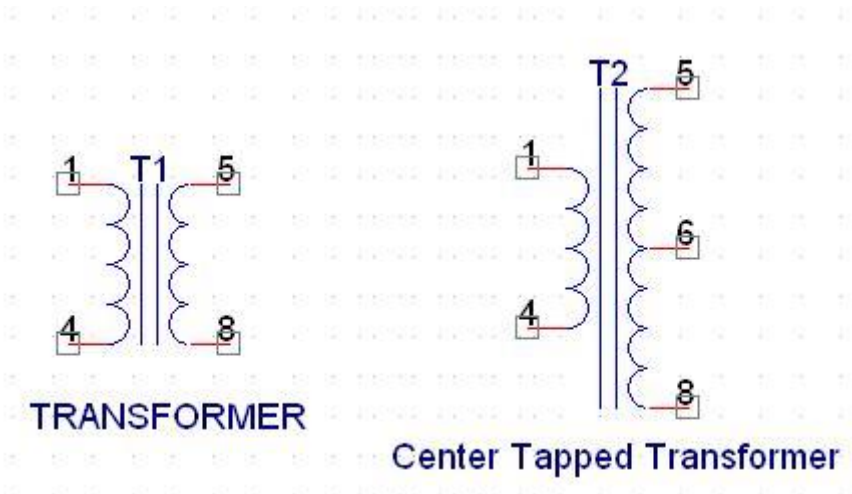


Fig No : 1.6 Transformer

A transformer consists of two coils also called as “WINDINGS” namely PRIMARY & SECONDARY. They are linked together through inductively coupled electrical conductors also called as CORE. A changing current in the primary causes a change in the Magnetic Field in the core & this in turn induces an alternating voltage in the secondary coil. If load is applied to the secondary then an alternating current will flow through the load. If we consider an ideal condition then all the energy from the primary circuit will be transferred to the secondary circuit through the magnetic field.

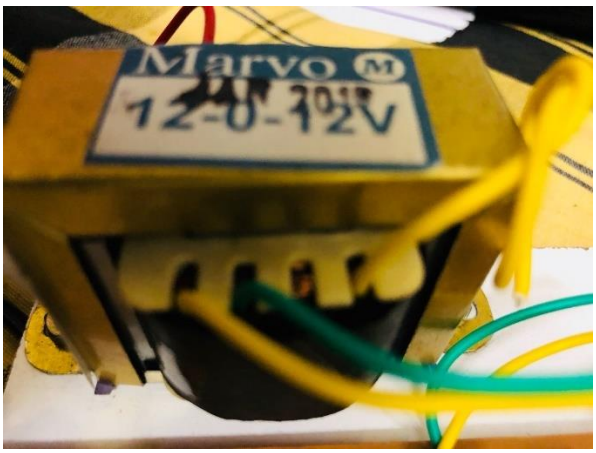


Fig No :1.6 Transformer

2.6.1 Rectifier

A rectifier is a device that converts an AC signal into DC signal. For rectification purpose we use a diode, a diode is a device that allows current to pass only in one direction i.e. when the anode of the diode is positive with respect to the cathode also called as forward biased condition & blocks current in the reversed biased condition.

Rectifier can be classified as follows:

1) Half Wave rectifier.

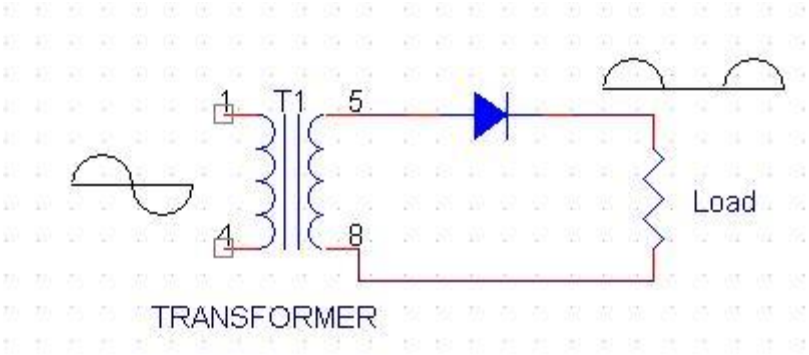


Fig No : 1.6.1 Half Wave rectifier

This is the simplest type of rectifier as you can see in the diagram a half wave rectifier consists of only one diode. When an AC signal is applied to it during the positive half cycle the diode is forward biased & current flows through it. But during the negative half cycle diode is reverse biased & no current flows through it. Since only one half of the input reaches the output, it is very inefficient to be used in power supplies.

2) Full wave rectifier.

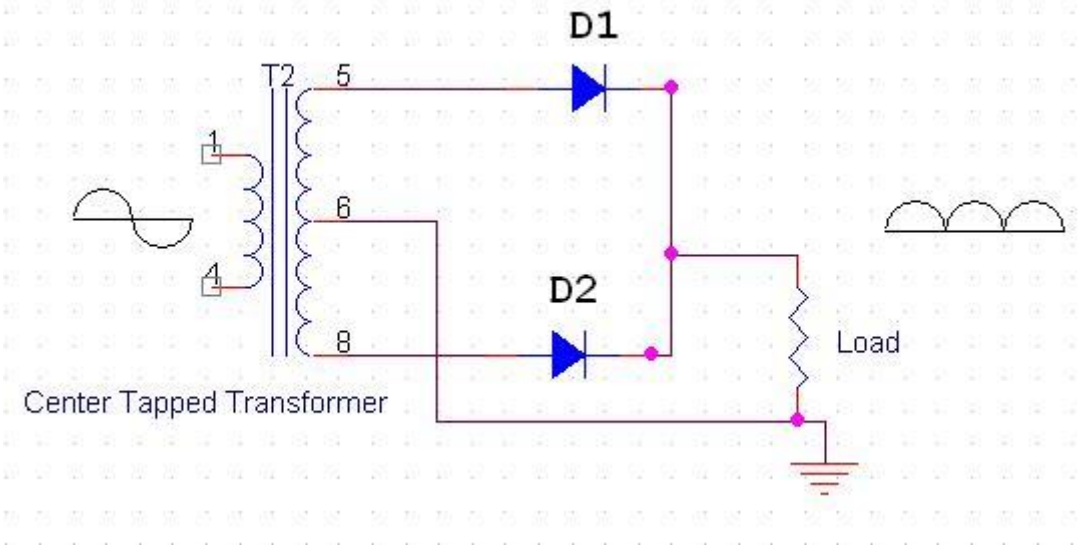


Fig no 1.6.2 Full wave rectifier

Half wave rectifier is quite simple but it is very inefficient, for greater efficiency we would like to use both the half cycles of the AC signal. This can be achieved by using

a center tapped transformer i.e. we would have to double the size of secondary winding & provide connection to the center. So during the positive half cycle diode D1 conducts & D2 is in reverse biased condition. During the negative half cycle diode D2 conducts & D1 is reverse biased. Thus we get both the half cycles across the load. One of the disadvantages of Full Wave Rectifier design is the necessity of using a center tapped transformer, thus increasing the size & cost of the circuit. This can be avoided by using the Full Wave Bridge Rectifier.

3) BridgeRectifier.

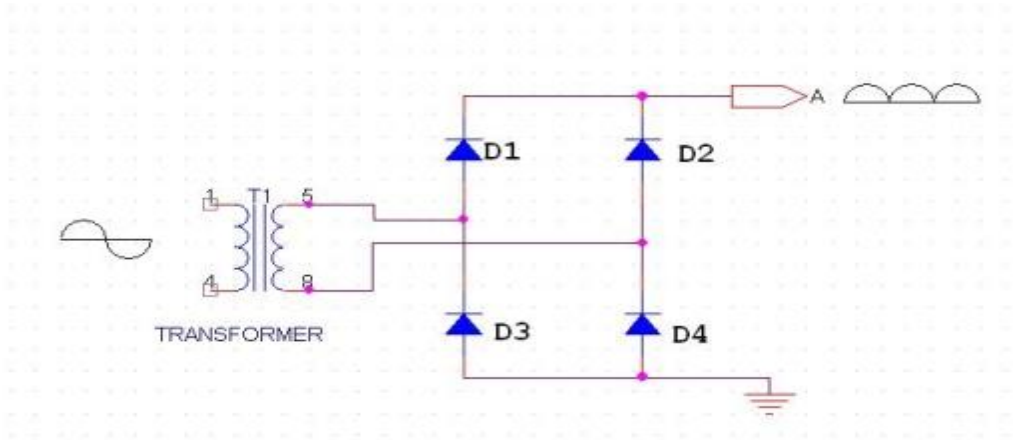


Fig no: 1.6.3 Bridge rectifier

As the name suggests it converts the full wave i.e. both the positive & the negative half cycle into DC thus it is much more efficient than Half Wave Rectifier & that too without using a center tapped transformer thus much more cost effective than Full Wave Rectifier.

Full Bridge Wave Rectifier consists of four diodes namely D1, D2, D3 and D4. During the positive half cycle diodes D1 & D4 conduct whereas in the negative half cycle diodes D2 & D3 conduct thus the diodes keep switching the transformer connections so we get positive half cycles in the output.

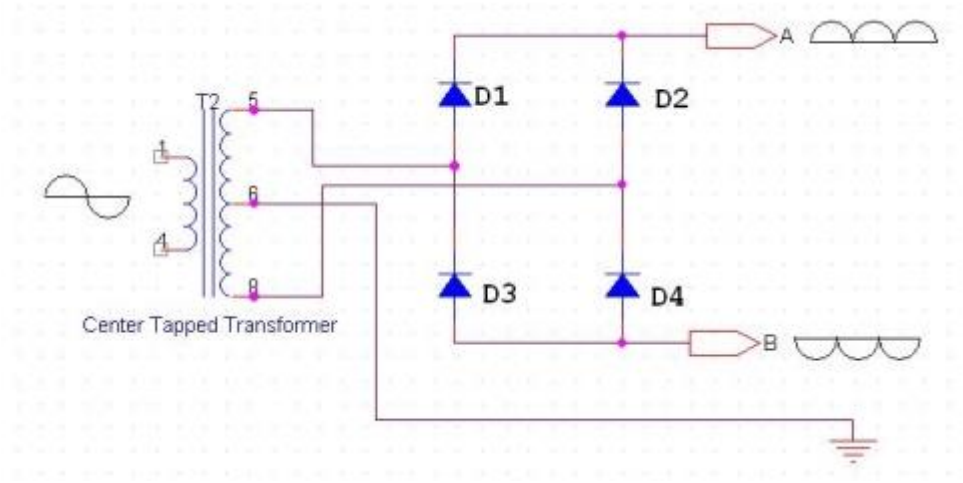


Fig no: 1.6.4 Center Tapped Transformer

2.6.2 Filter Capacitor

Even though half wave & full wave rectifier give DC output, none of them provides a constant output voltage. For this we require to smoothen the waveform received from the rectifier. This can be done by using a capacitor at the output of the rectifier this capacitor is also called as “FILTER CAPACITOR” or “SMOOTHING CAPACITOR” or “RESERVOIR CAPACITOR”. Even after using this capacitor a small amount of ripple will remain.

We place the Filter Capacitor at the output of the rectifier the capacitor will charge to the peak voltage during each half cycle then will discharge its stored energy slowly through the load while the rectified voltage drops to zero, thus trying to keep the voltage as constant as possible.

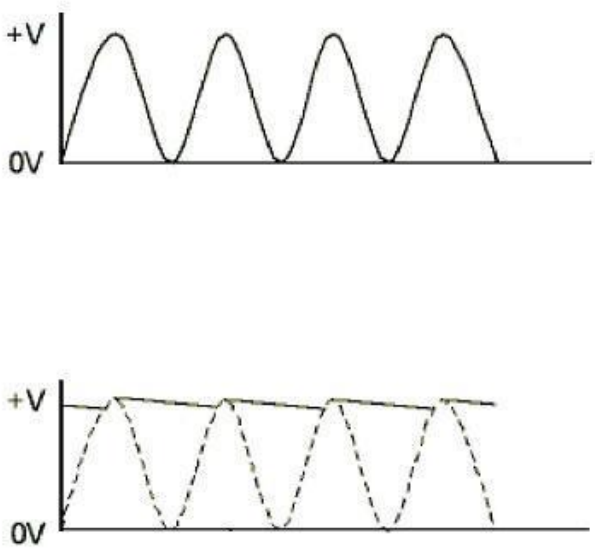


Fig No: 1.6.5 Filter capacitor

If we go on increasing the value of the filter capacitor then the Ripple will decrease. But then the costing will increase. The value of the Filter capacitor depends on the current consumed by the circuit, the frequency of the waveform & the accepted ripple.

$$C = \frac{V_r F}{I}$$

Where,

Vr= accepted ripple voltage.(should not be more than 10% of the voltage)

I= current consumed by the circuit in Amperes.

F= frequency of the waveform. A half wave rectifier has only one peak in one cycle so F=25hz Whereas a full wave rectifier has Two peaks in one cycle so F=100hz.

2.6.3 VOLTAGE REGULATOR

A Voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types

1) Linear Voltage Regulator

Also called as Resistive Voltage regulator because they dissipate the excessive voltage resistively as heat.

2) Switching Regulators.

They regulate the output voltage by switching the Current ON/OFF very rapidly. Since their output is either ON or OFF it dissipates very low power thus achieving higher efficiency as compared to linear voltage regulators. But they are more complex & generate high noise due to their switching action. For low level of output power switching regulators tend to be costly but for higher output wattage they are much cheaper than linear regulators.

The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.

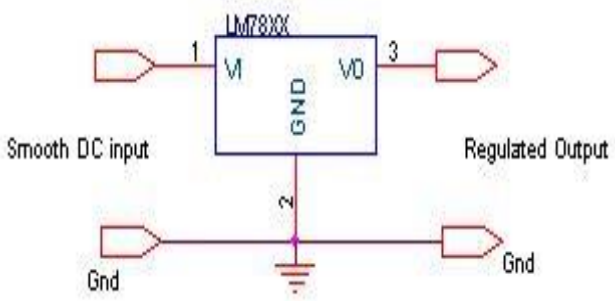


Fig No :1.6.6 Voltage Regulator

After filtering the rectifier output the signal is given to a voltage regulator. The maximum input voltage that can be applied at the input is 35V.Normally there is a 2-3 Volts drop across the regulator so the input voltage should be at least 2-3 Volts higher than the output voltage. If the input voltage gets below the Vmin of the regulator due to the ripple voltage or due to any other reason the voltage regulator will not be able to produce the correct regulated voltage.

(i) 3 Circuit diagram:

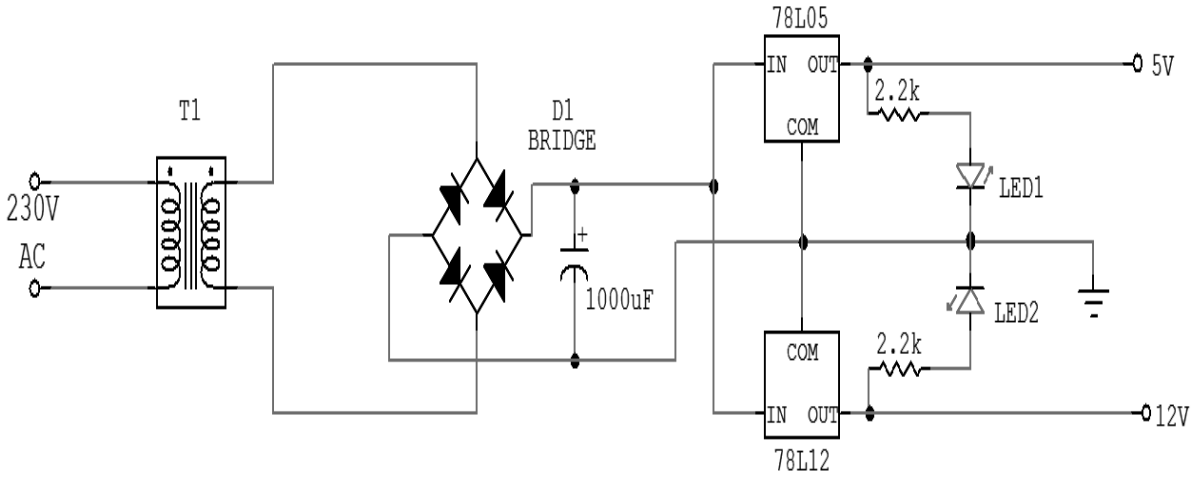


Fig No : 1.6.7 Circuit Diagram of power supply

(ii) IC 7805:

7805 is an integrated three-terminal positive fixed linear voltage regulator. It supports an input voltage of 10 volts to 35 volts and output voltage of 5 volts. It has a current rating of 1 amp although lower current models are available. Its output voltage is fixed at 5.0V. The 7805 also has a built-in current limiter as a safety feature. 7805 is manufactured by many companies, including National Semiconductors and Fairchild Semiconductors.

The 7805 will automatically reduce output current if it gets too hot.The last two digits represent the voltage; for instance, the 7812 is a 12-volt regulator. The 78xx series of regulators is designed to work in complement with the 79xx series of negative voltage regulators in systems that provide both positive and negative regulated voltages, since the 78xx series can't regulate negative voltages in such a system. The 7805 & 78 is one of the most common and well-known of the 78xx series regulators, as it's small component count and medium-power regulated 5V make it useful for powering TTL devices.

SPECIFICATIONS	IC 7805
V_{out}	5V
V_{ein} - V_{out} Difference	5V - 20V

Operation Ambient Temp	0 - 125°C
Output I_{max}	1A

Table 2: Specifications of IC7805

2.7 LCD MODULE

To display interactive messages we are using LCD Module. We examine an intelligent LCD display of two lines,16 characters per line that is interfaced to the controllers. The protocol (handshaking) for the display is as shown. Whereas D0 to D7th bit is the Data lines, RS, RW and EN pins are the control pins and remaining pins are +5V, -5V and GND to provide supply. Where RS is the Register Select, RW is the Read Write and EN is the Enable pin.

The display contains two internal byte-wide registers, one for commands (RS=0) and the second for characters to be displayed (RS=1). It also contains a user-programmed RAM area (the character RAM) that can be programmed to generate any desired character that can be formed using a dot matrix. To distinguish between these two data areas, the hex command byte 80 will be used to signify that the display RAM address 00h will be chosen.Port1 is used to furnish the command or data type, and ports 3.2 to3.4 furnish register select and read/write levels.The display takes varying amounts of time to accomplish the functions as listed. LCD bit 7 is monitored for logic high (busy) to ensure the display is overwritten.

Liquid Crystal Display also called as LCD is very helpful in providing user interface as well as for debugging purpose. The most common type of LCD controller is HITACHI 44780 which provides a simple interface between the controller & an LCD. These LCD's are very simple to interface with the controller as well as are cost effective.

Fig No 1.7 2x16 Line Alphanumeric LCD Display

The most commonly used ALPHANUMERIC displays are 1x16 (Single Line & 16 characters), 2x16 (Double Line & 16 character per line) & 4x20 (four lines & Twenty characters per line). The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number on data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required.

Pin	Symbol	Function
1	Vss	Ground
2	Vdd	Supply Voltage
3	Vo	Contrast Setting
4	RS	Register Select
5	R/W	Read/Write Select
6	En	Chip Enable Signal
7-14	DB0-DB7	Data Lines
15	A/Vee	Gnd for the backlight
16	K	Vcc for backlight

Table:3 LCD symblos

When *RS* is low (0), the data is to be treated as a command. When RS is high (1), the data being sent is considered as text data which should be displayed on the screen.

When *R/W* is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively reading from the LCD. Most of the times there is no need to read from the LCD so this line can directly be connected to Gnd thus saving one controller line.The *ENABLE* pin is used to latch the data present on the data pins. A HIGH - LOW signal is required to latch the data. The LCD interprets and executes our command at the instant the EN line is brought low. If you never bring EN low, your instruction will never be executed.

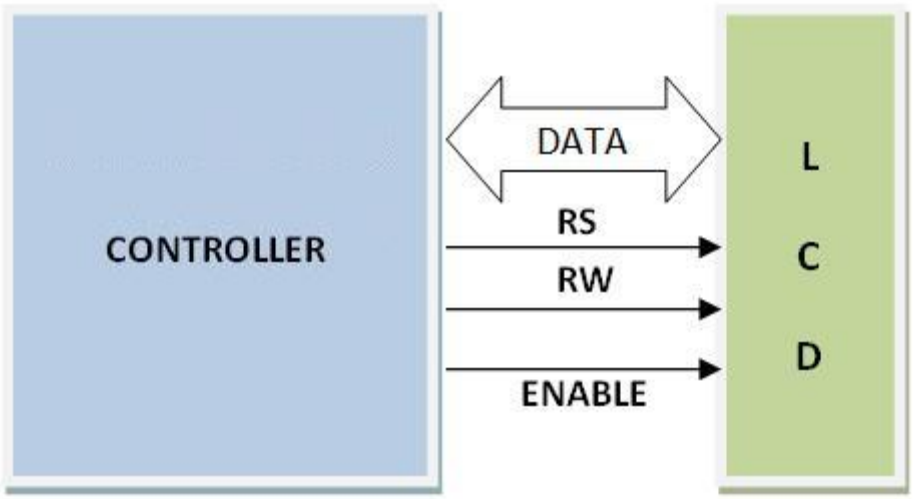


Fig No : 1.7.1 LCD display pins

COMMANDS USED IN LCD

No.	Instruction	Hex	Decimal
1	Function Set: 8-bit, 1 Line, 5x7 Dots	0x30	48
2	Function Set: 8-bit, 2 Line, 5x7 Dots	0x38	56
3	Function Set: 4-bit, 1 Line, 5x7 Dots	0x20	32
4	Function Set: 4-bit, 2 Line, 5x7 Dots	0x28	40
5	Entry Mode	0x06	6
6	Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
7	Display on Cursor on	0x0E	14
8	Display on Cursor off	0x0C	12
9	Display on Cursor blinking	0x0F	15
10	Shift entire display left	0x18	24
12	Shift entire display right	0x1C	30
13	Move cursor left by one character	0x10	16
14	Move cursor right by one character	0x14	20
15	Clear Display (also clear DDRAM content)	0x01	1

. **Fig No : 1.7.2 Commands for LCD**

2.8 GSM Module

The words, “Mobile Station” (MS) or “Mobile Equipment” (ME) are used for mobile terminals

Supporting GSM services.A call from a GSM mobile station to the PSTN is called a “mobile originated call” (MOC) or“Outgoing call”, and a call from a fixed network to a GSM mobile station is called a “mobile Terminated call” (MTC) or “incoming call”.

Fig No 1.8 GSM Module

2.8.1 What is GSM?

GSM (Global System for Mobile communications) is an open, digital cellular technology used for transmitting mobile voice and data services.

What does GSM offer?

GSM supports voice calls and data transfer speeds of up to 9.6 kbit/s, together with the transmission of SMS (Short Message Service).

GSM operates in the 900MHz and 1.8GHz bands in Europe and the 1.9GHz and 850MHz bands in the US. The 850MHz band is also used for GSM and 3G in Australia, Canada and many South American countries. By having harmonised spectrum across most of the globe, GSM's international roaming capability allows users to access the same services when travelling abroad as at home. This gives consumers seamless and same number connectivity in more than 218 countries. Terrestrial GSM networks now cover more than 80% of the world's population. GSM satellite roaming has also extended service access to areas where terrestrial coverage is not available.

2.8.2 HISTORY

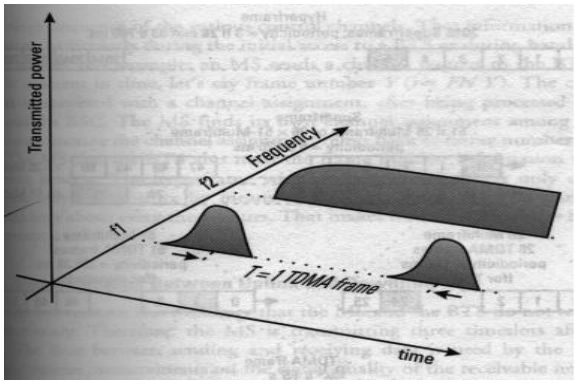
In 1980's the analog cellular telephone systems were growing rapidly all throughout Europe, France and Germany. Each country defined its own protocols and frequencies to work on. For example UK used the Total Access Communication System (TACS), USA used the AMPS technology and Germany used the C-netz technology. None of these systems were interoperable and also they were analog in nature.

In 1982 the Conference of European Posts and Telegraphs (CEPT) formed a study group called the GROUPE SPECIAL MOBILE (GSM) The main area this

focused on was to get the cellular system working throughout the world, and ISDN compatibility with the ability to incorporate any future enhancements. In 1989 the GSM transferred the work to the European Telecommunications Standards Institute (ETSI.) the ETS defined all the standards used in GSM.

2.8.3 BASICS OF WORKING AND SPECIFICATIONS OF GSM :

The GSM architecture is nothing but a network of computers. The system has to partition available frequency and assign only that part of the frequency spectrum to any base transreceiver station and also has to reuse the scarce frequency as often as possible. GSM uses TDMA and FDMA together. Graphically this can be shown below –



2.8.4 Technical specifications of GSM

Multiple Access Method	TDMA / FDMA
Uplink frequencies (MHz)	933-960 (basic GSM)
Downlink frequencies (MHz)	890-915 (basic GSM)
Duplexing	FDD
Channel spacing, kHz	200
Modulation	GMSK
Portable TX power, maximum / average (mW)	1000 / 125
Power control, handset and BSS	Yes
Speech coding and rate (kbps)	RPE-LTP / 13
Speech Channels per RF channel:	8
Channel rate (kbps)	270.833
Channel coding	Rate 1/2 convolutional
Frame duration (ms)	4.615

GSM was originally defined for the 900 Mhz range but after some time even the 1800 Mhz range was used for cellular technology. The 1800 MHz range has its architecture and specifications almost same to that of the 900 Mhz GSM technology

but building the Mobile exchanges is easier and the high frequency Synergy effects add to the advantages of the 1800 Mhz range.

2.8.5 ARCITECTURE AND BUILDIGN BLOCKS

- GSM Radio Network – This is concerned with the signaling of the system. Hand-overs occur in the radio network. Each BTS is allocated a set of frequency channels.
- GSM Mobile switching Network – This network is concerned with the storage of data required for routing and service provision.
- GSM Operation and Maintenance – The task carried out by it include Administration and commercial operation , Security management, Network configuration, operation, performance management and maintenance tasks.

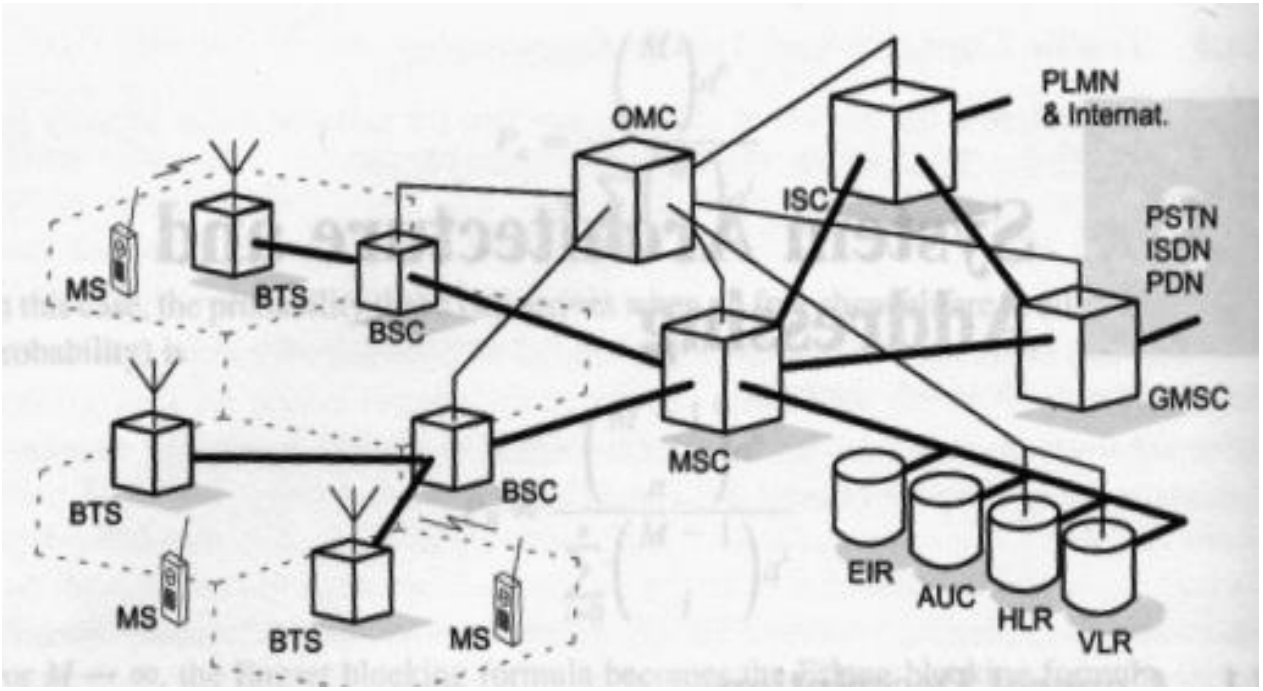


Fig No 1.8.1 GSM BLOCK

2.8.6 SIGNALLING SCHEMES AND CIPHERING CODES USED –

GSM is digital but voice is inherently analog. So the analog signal has to be converted and then transmitted. The coding scheme used by GSM is RPE-LTP (Rectangular pulse Excitation – Long Term Prediction)

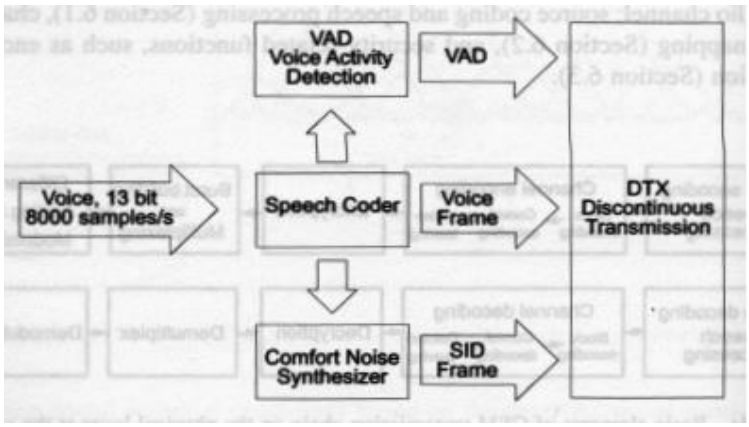


Fig No 1.8.2 Transmitter for the voice signal

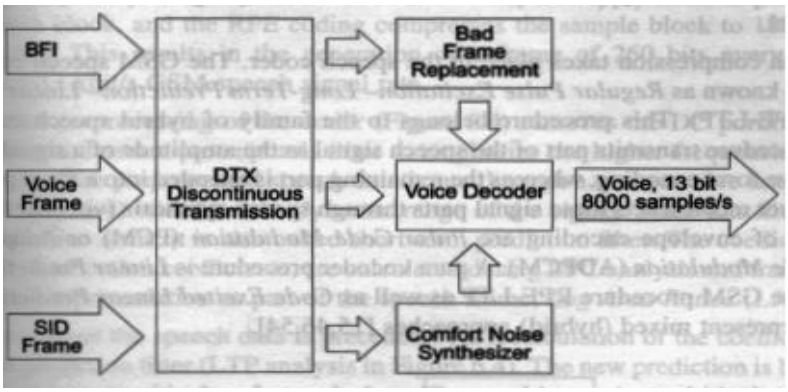


Fig.No 1.8.3 Receiver for the Voice signal

The voice signal is sampled at 8000 bits/sec and is quantized to get a 13 bit resolution corresponding to a bit rate of 104 kbits/sec. This signal is given to a speech coder (codec) that compresses this speech into a source-coded speech signal of 260 bit blocks at a bit rate of 13 kbit/sec. The codec achieves a compression ratio of 1:8. The coder also has a Voice activity detector (VAD) and comfort noise synthesizer. The VAD decides whether the current speech frame contains speech or pause, this is turn is used to decide whether to turn on or off the transmitter under the control of the Discontinuous Transmission (DTX). This transmission takes advantage of the fact that during a phone conversation both the parties rarely speak.

Thus the DTX helps in reducing the power consumption and prolonging battery life. The missing speech frames are replaced by synthetic background noise generated by the comfort noise synthesize in a Silence Descriptor (SID) frame. Suppose a loss off speech frame occurs due to noisy transmission and it cannot be corrected by the channel coding protection mechanism then the decoder flags such frames with a bad frame indicator (BFI) In such a case the speech frame is discarded and using a technique called error concealment which calculates the next frame based on the previous frame.

2.8.7 Multi-Tech line settings

A serial link handler is set with the following default values (factory settings): autobaud, 8 bits data, 1 stop bit, no parity, RTS/CTS flow control. Please use the +IPR, +IFC and +ICFCommands to change these settings. Commands always start with AT (which means ATtention) and finish with a <CR> character. Information responses and result codes Responses start and end with <CR><LF>, except for the ATV0 DCE response format) and the ATQ1 (result code suppression) commands.If command syntax is incorrect, an ERROR string is returned.If command syntax is correct but with some incorrect parameters, the +CME ERROR: <Err> or +CMS ERROR: <SmsErr> strings are returned with different error codes. If the command line has been performed successfully, an OK string is returned.In some cases, such as "AT+CPIN?" or (unsolicited) incoming events, the product does not return the OK string as a response.

Product Serial Number +CGSN

This command allows the user application to get the IMEI (International Mobile Equipment

Identity) of the product.

Syntax:

Command syntax: AT+CGSN

Command	Possible responses
AT+CGSN *Note: Get the IMEI*	**012345678901234** OK *Note: IMEI read from EEPROM*
AT+CGSN *Note: Get the IMEI*	+CME ERROR: 22 *Note: IMEI not found in EEPROM*

Table: 4 Serial Number

Repeat last command A/

This command repeats the previous command. Only the A/ command itself cannot be repeated.

Syntax:

Command syntax: A/

Command	Possible responses
A/ *Note: Repeat last command*	

Table:5 Last Command

Signal Quality +CSQ

This command determines the received signal strength indication (<rssi>) and the channel bit error rate (<ber>) with or without a SIM card inserted.

Syntax:

Command syntax: AT+CSQ

Command	Possible responses
AT+CSQ	+CSQ: <rssi>,<ber> OK *Note: <rssi> and <ber> as defined below*

Table:6 Signal Quality

Defined values:

0: -113 dBm or less

1: -111 dBm

30: -109 to –53 dBm

31: -51dBm or greater

99: not known or not detectable

<ber>: 0…7: as RXQUAL values in the table GSM 05.08

99: not known or not detectable

New message indication +CNMI

This command selects the procedure for message reception from the network.

Syntax:

Command syntax: AT+CNMI=<mode>,<mt>,<bm>,<ds>,<bfr>

Command	Possible responses
AT+CNMI=2,1,0,0,0 *Note: <mt>=1*	OK
	AT+CMTI: "SM",1 *Note: message received*
AT+CNMI=2,2,0,0,0 *Note: <mt>=2*	OK
	+CMT: "123456","98/10/01,12:30 00+00",129,4 ,32,240, "15379",129,5<CR><LF> message received *Note: message received*
AT+CNMI=2,0,0,1,0 *Note: <ds>=1*	OK
Message to send <ctrl-Z> *Note: Send a message in text mode*	+CMGS: 7 OK *Note: Successful* AT+CMGS="+33146290800"<CR> *transmission*
	+CDS: 2, 116, "+33146290800", 145, "98/10/01,12:30:07+04", "98/10/01 12:30:08+04", 0 *Note: message was correctly delivered*

Table: 7 New Message Indication

Read message +CMGR

This command allows the application to read stored messages. The messages are read from the

memory selected by **+CPMS** command.

Command syntax: AT+CMGR=<index>

Command	Possible responses
	AT+CMTI: "SM",1 *Note: New message received*
AT+CMGR=1 *Note: Read the message*	+CMGR: "REC UNREAD","0146290800", "98/10/01,18:22:11+00",<CR><LF> ABCdefGHI OK
AT+CMGR=1 *Note: Read the message again*	+CMGR: "REC UNREAD","0146290800", "98/10/01,18:22:11+00",<CR><LF> ABCdefGHI OK *Note: Message is read now*
AT+CMGR=2 *Note: Read at a wrong index*	+CMS ERROR: 321 *Note: Error: invalid index*
AT+CMGF=0 ;+CMGR=1 *Note: In PDU mode*	+CMGR: 2,,<length> <CR><LF><pdu> OK *Note: Message is stored but unsent, no <alpha>field*

Table: 8 Read Message

List message +CMGL

This command allows the application to read stored messages, by indicating the type of the

Message to read. The messages are read from the memory selected by the +CPMS command.

Syntax: Command syntax: AT+CMGL=<stat>

Command	Possible responses
AT+CMGL="REC UNREAD" *Note: List unread messages in text mode*	+CMGL: 1,"REC UNREAD","0146290800", <CR><LF> Unread message ! +CMGL: 3,"REC UNREAD", "46290800", <CR><LF> Another message unread! OK *Note: 2 messages are unread, these messages will then have their status changed to "REC READ" (+CSDH:0)*
AT+CMGL="REC READ" *Note: List read messages in text mode*	+CMGL: 2,"REC READ","0146290800", <CR><LF> Keep cool OK
AT+CMGL="STO SENT" *Note: List stored and sent messages in text mode*	OK *Note: No message found*
AT+CMGL=1 *Note: List read messages in PDU mode*	+CMGL: 1,1,,26 <CR><LF> 07913366003000F3040B913366920547F4001300119041253 0400741AA8E5A9C5201 OK

Table: 9 List of Messages

Send message +CMGS

The <address> field is the address of the terminal to which the message is sent. To send the Message, simply type, <ctrl-Z> character (ASCII 26). The text can contain all existing

Characters except <ctrl-Z> and <ESC> (ASCII 27). This command can be aborted using the

<ESC> character when entering text. In PDU mode, only hexadecimal characters are used

('0'…'9','A'…'F').

Syntax:

Command syntax in text mode*:*

AT+CMGS= <da> [,<toda>] <CR>

text is entered <ctrl-Z / ESC >

Command	Possible responses
AT+CMGS="+33146290800"<CR> Please call me soon, Fred. <ctrl-Z> *Note: Send a message in text mode*	+CMGS: <mr> OK *Note: Successful transmission*
AT+CMGS=<length><CR><pdu><ctrl-Z> *Note: Send a message in PDU mode*	+CMGS: <mr> OK *Note: Successful transmission*

Table: 10 Sending Messages

The message reference, <mr>, which is returned to the application, is allocated by the product.

This number begins with 0 and is incremented by one for each outgoing message (successful

and failure cases); it is cyclic on one byte (0 follows 255).

2.9 GPS GLOBAL POSITIONING SYSTEM)

The Global Positioning System (GPS) is a U.S. space-based radio navigation system that provides reliable positioning, navigation, and timing services to civilian users on a continuous worldwide basis -- freely available to all. For anyone with a GPS receiver, the system will provide location and time. GPS provides accurate location and time information for an unlimited number of people in all weather, day and night, anywhere in the world.

The GPS is made up of three parts:

1. Satellites orbiting the Earth

2. Control and monitoring stations on Earth

3. The GPS receivers owned by users.

GPS satellites broadcast signals from space that are picked up and identified by GPS receivers. Each GPS receiver then provides three-dimensional location (latitude, longitude, and altitude) plus the time.

1. **SPACE SEGMENT**
    - 24+ satellites

- 20,200 km altitude
- 55 degrees inclination
- 12 hour orbital period
- 5 ground control stations
- Each satellite passes over a ground monitoring station every 12 hours

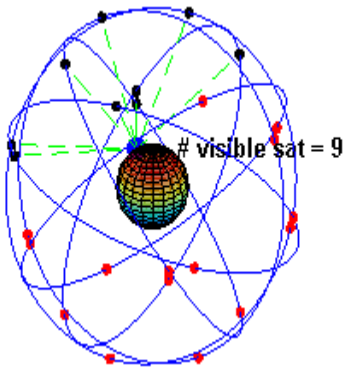


Fig No : 1.9 GPS Satellite

The space segment is composed of the orbiting GPS satellites or Space Vehicles (SV) in GPS parlance. The GPS design originally called for 24 SVs, this was modified to six planes with four satellites each. The orbital planes are centered on the Earth, not rotating with respect to the distant stars. The six planes have approximately 55° inclination (tilt relative to Earth's equator) and are separated by 60° right ascension of the ascending node (angle along the equator from a reference point to the orbit's intersection). The orbits are arranged so that at least six satellites are always within line of sight from almost everywhere on Earth's surface.

The full constellation of 24 satellites that make up the GPS space segment are orbiting the earth about 20,200 km above us. They are constantly moving, making two complete orbits in less than 24 hours. These satellites are travelling at speeds of roughly 7,000 miles an hour.GPS satellites are powered by solar energy. They have backup batteries onboard to keep them running in the event of a solar eclipse, when there's no solar power. Small rocket boosters on each satellite keep them flying in the correct path.Here are some other interesting facts about the GPS satellites (also called NAVSTAR, the official U.S. Department of Defense name for GPS):

- The first GPS satellite was launched in 1978.
- A full constellation of 24 satellites was achieved in 1994.
- Each satellite is built to last about 10 years. Replacements are constantly being built and launched into orbit.
- A GPS satellite weighs approximately 2,000 pounds and is about 17 feet across with the solar panels extended.
- Transmitter power is only 50 watts or less.

The receiver must be aware of the PRN codes for each satellite to reconstruct the actual message data. The C/A code, for civilian use, transmits data at 1.023 million chips per second, whereas the P code, for U.S. military use, transmits at 10.23 million chips per second. The L1 carrier is modulated by both the C/A and P codes, while the L2 carrier is only modulated by the P code. The P code can be encrypted as a so-called P(Y) code which is only available to military equipment with a proper decryption key. Both the C/A and P(Y) codes impart the precise time-of-day to the user

2.Control and monitoring stations on Earth

Ground Stations (also known as the "Control Segment")These stations monitor the GPS satellites, checking both their operational health and their exact position in space. The master ground station transmits corrections for the satellite's ephemeris constants and clock offsets back to the satellites themselves. The satellites can then incorporate these updates in the signals they send to GPS receivers. There are five monitor stations: Hawaii, Ascension Island, Diego Garcia, Kwajalein, and Colorado Springs.

Each GPS satellite regularly with a navigational update using dedicated or shared ground antennas (GPS dedicated ground antennas are located at Kwajalein, Ascension Island, Diego Garcia, and Cape Canaveral). These updates synchronize the atomic clocks on board the satellites to within a few nanoseconds of each other, and adjust the ephemeris of each satellite's internal orbital model. The updates are created by a Kalman filter, which uses inputs from the ground monitoring stations, space weather information, and various other inputs. Satellite maneuvers are not precise by GPS

standards. So to change the orbit of a satellite, the satellite must be marked unhealthy, so receivers will not use it in their calculation. Then the maneuver can be carried out, and the resulting orbit tracked from the ground. Then the new ephemeris is uploaded and the satellite marked healthy again.

The GPS receivers

- Receiver determines location, speed, direction, and time
- 3 satellite signals are necessary to locate the receiver in 3D space
- 4th satellite is used for time accuracy
- Position calculated within sub-centimeter scale

Individuals may purchase GPS handsets that are readily available through commercial retailers. Equipped with these GPS receivers, users can accurately locate where they are and easily navigate to where they want to go, whether walking, driving, flying, or boating.

Today's GPS receivers are extremely accurate, thanks to their parallel multi-channel design. Garmin's 12 parallel channel receivers are quick to lock onto satellites when first turned on and they maintain strong locks, even in dense foliage or urban settings with tall buildings. Certain atmospheric factors and other sources of error can affect the accuracy of GPS receivers. Garmin® GPS receivers are accurate to within 15 meters on average.

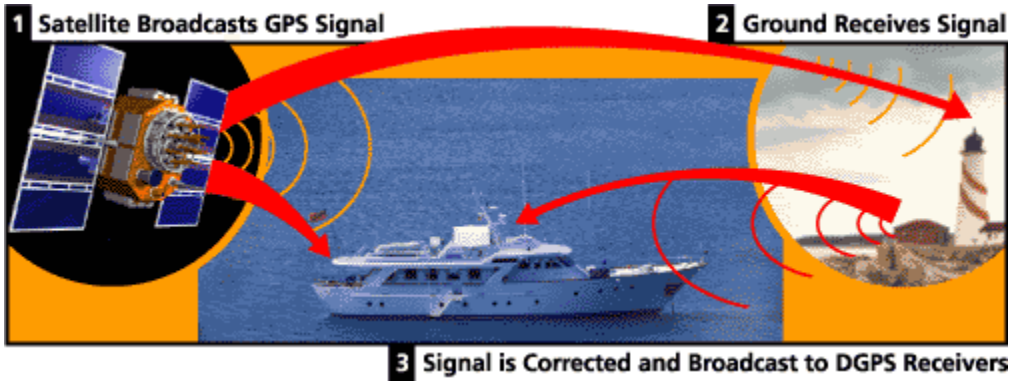


Fig No : 1.9.1 GPS receiver

2.9.2 COMMANDS IN GPS

NMEA record	Description
GGA	Global positioning system fixed data
GLL	Geographic position - latitude/longitude
GSA	GNSS DOP and active satellites
GSV	GNSS satellites in view
RMC	Recommended minimum specific GNSS data
VTG	Course over ground and ground speed

Table : 11 Commands in GPS

2.10 WIFI MODULE(ESP)

The **ESP8266** is a low-cost Wi-Fi chip with full TCP/IP stack and MCU (Micro Controller Unit) capability produced by Shanghai-based Chinese manufacturer,.The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer, AI-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands. However, at the time there was almost no English-language documentation on the chip and the commands it accepted.

The very low price and the fact that there were very few external components on the module which suggests that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.The **ESP8285** is an ESP8266 with 1 MB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi.The successor to these module(s) is ESP32.This is the series of ESP8266-based modules made by Espressif.

AI-Thinker module

Fig No : 1.10 ESP-01 module

These are the first series of modules made with the ESP8266 by the third-party manufacturer *AI-Thinker* and remain the most widely available. They are collectively referred to as "ESP-xx modules". To form a workable development system they require additional components, especially a serial TTL-to-USB adapter (sometimes called a USB-to-UART bridge) and an external 3.3 Volt power supply. Novice ESP-8266 developers are encouraged to consider larger ESP8266.

Wi-Fi development boards like the Node MCU which includes the USB-to-UART bridge and a Micro-USB connector coupled with a 3.3 Volt power regulator already built into the board. When project development is complete, you may not need these components and can consider using these cheaper ESP-xx modules as a lower power, smaller footprint option for your production runs.ESP8266 offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

When ESP8266 hosts the application, and when it is the only application processor in the device, it is able to boot up directly from an external flash. It has integrated cache to improve the performance of the system in such applications, and Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any microcontroller-based design with simple connectivity through UART interface or the CPU AHB bridge interface.

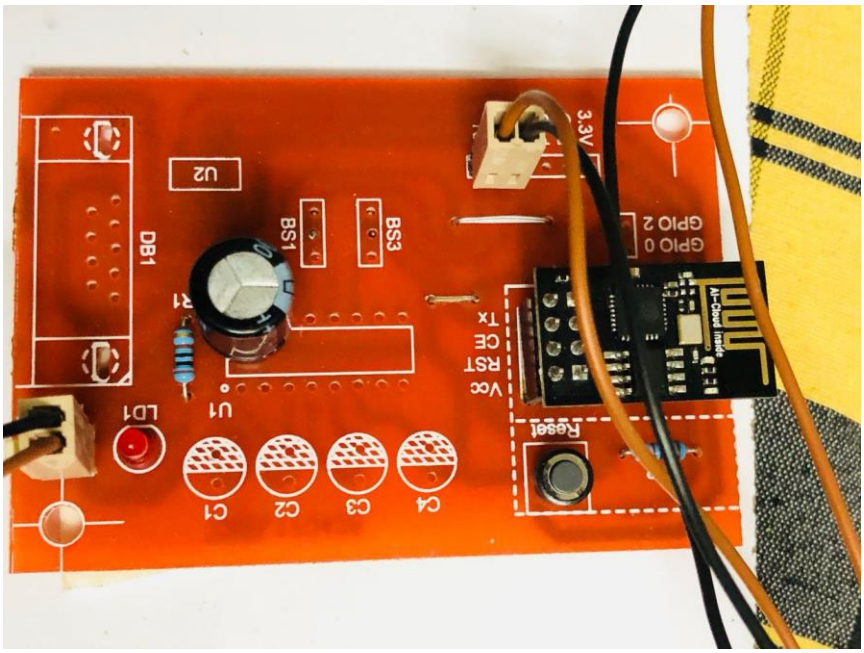


Fig No :1.11 WIFI Module

2.11 Ultrasonic Sensor

This "ECHO" Ultrasonic Distance Sensor from Rhydolabz is an amazing product that provides very short (2CM) to long-range (4M) detection and ranging. The sensor provides precise, stable non- contact distance measurements from 2cm to 4 meters with very high accuracy. Its compact size, higher range and easy usability make it a handy sensor for distance measurement and mapping. The board can easily be interfaced to microcontrollers where the triggering and measurement can be done using one I/O pin. The sensor transmits an ultrasonic wave and produces an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.

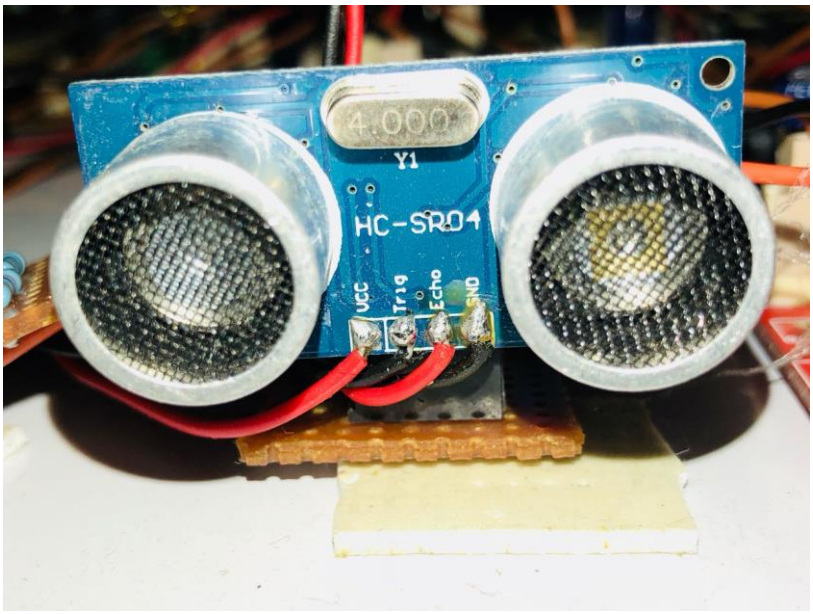


Fig No 1.12 Ultrasonic Sensor

FEATURES

- Professional EMI/RFI Complaint PCB Layout Design for Noise Reduction
- Range : 2 cm to 4 m
- Accurate and Stable range data
- Data loss in Error zone eliminated
- Modulation at 40 KHz
- Mounting holes provided on the circuit board
- Triggered externally by supplying a pulse to the signal pin
- 5V DC Supply voltage
- Current - < 20mA
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 VTTL or 3.3V CMOS microcontrollers
- Echo pulse: positive TTL pulse, 87 ^s minimum to 30 ms maximum(PWM)
- On Board Burst LED Indicator shows measurement in progress

PIN DEFINITION

PIN	PIN NAME	DETALES
VCC	Power Supply	Power Supply Input (+5V)
GND	Ground	Ground Level of Power supply
SIGNAL	Signal I/O	This pin reads the trigger pulse from the host microcontroller and returns the pulse based on the distance.

Table :12 Pins for Ultrasonic sensor

2.12 Water Sensor

LM 358 consist of two independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltage. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifier, DC gain blocks and all the conventional OP-AMP circuits which now can be easily implemented in single power supply systems.

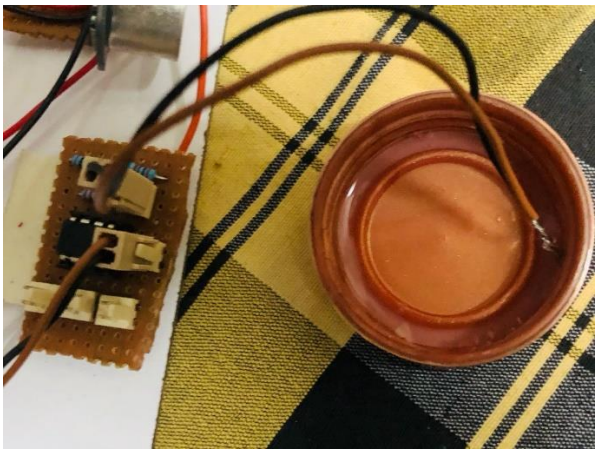


Fig No 1.13 Water Sensor

Features:

- Internally Frequency Compensated for Unity Gain
- Large DC Voltage Gain: 100dB
- Wide Power Supply Range
- LM358:
- Singal supply:3V~32V
- Or dual supply: ±1.5V~ 16V
- Input Common Mode Voltage Range Includes Ground
- Large Output Voltage Swing: 0V DC to Vcc -1.5V DC

2.13 MEMS Sensor

A flight data recorder (FDR) (also ADR, for accident data recorder) is an electronic device employed to record any instructions sent to any electronic systems on an aircraft. It is a device used to record specific aircraft performance parameters. Another kind of flight recorder is the cockpit voice recorder (CVR), which records conversation in the cockpit, radio communications between the cockpit crew and others (including conversation with air traffic control personnel), as well as ambient sounds. In some cases, both functions have been combined into a single unit Popularly referred to as a "**BLACK BOX**" the data recorded by the FDR is used for accident investigation, as well as for analyzing air safety issues, material degradation and engine performance.

Due to their importance in investigating Contrary to the "black box" reference, the exterior of the FDR is coated with heat-resistant bright Red paint for high visibility in wreckage, and the unit is usually mounted in the aircraft's tail section, where it is more likely to survive a severe crash. An accelerometer (MEMS) can be used in a alarm application in which predict before the crash . It can be used as a crash recorder of the flight movements before, during and after a crash.

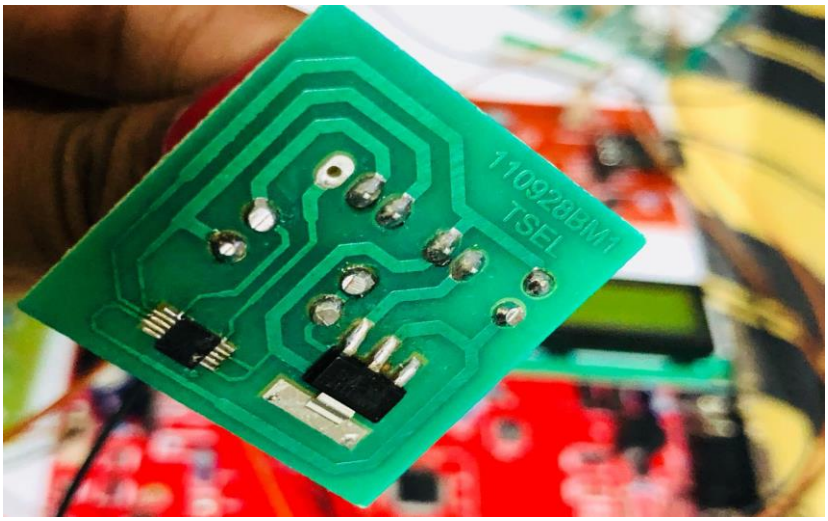


Fig:1.14 mems sensor

3. CIRCUITS DIAGRAM

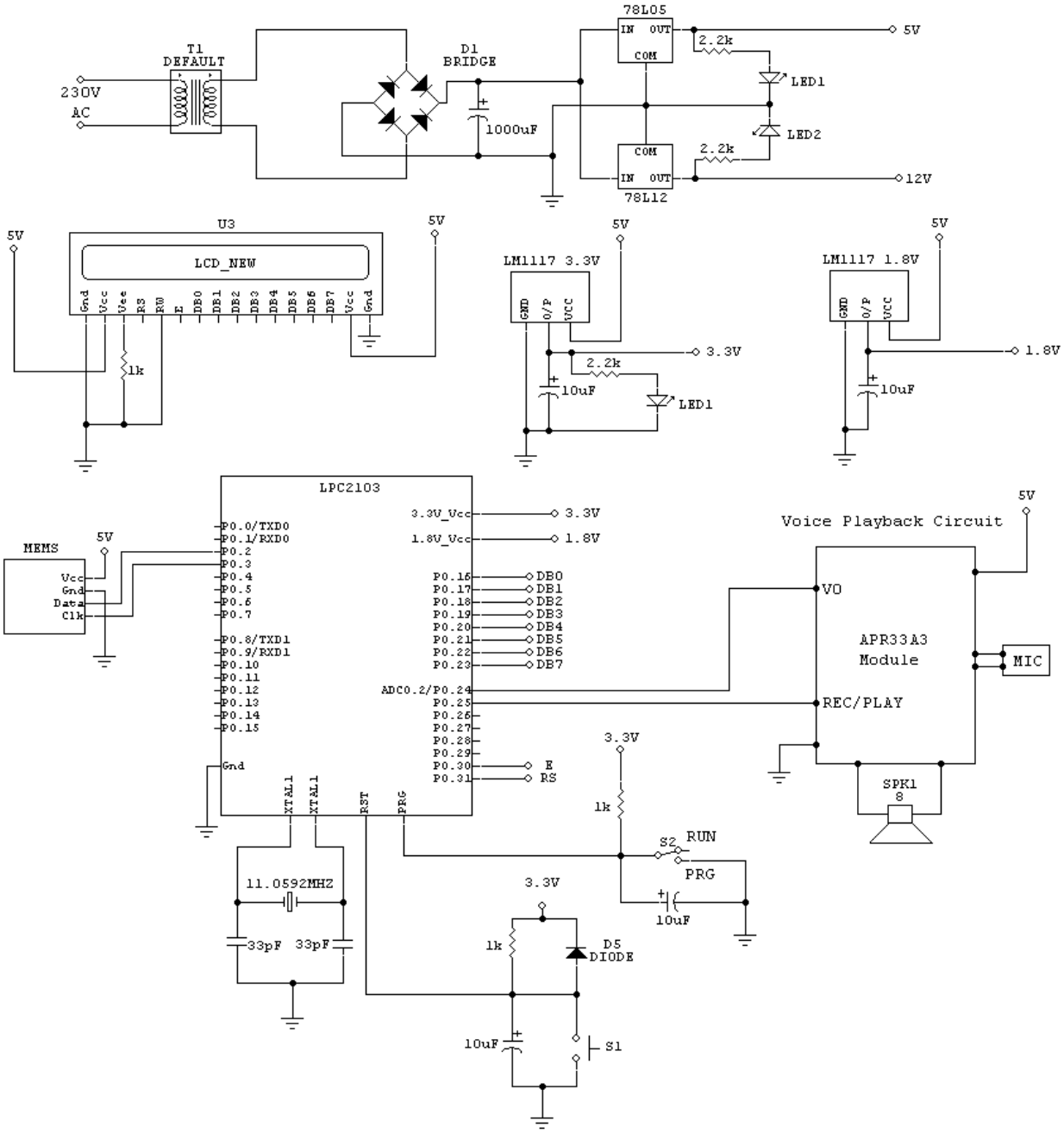


Fig No : 2 Circuit Diagram

4.Architecture

4.1 system architecture

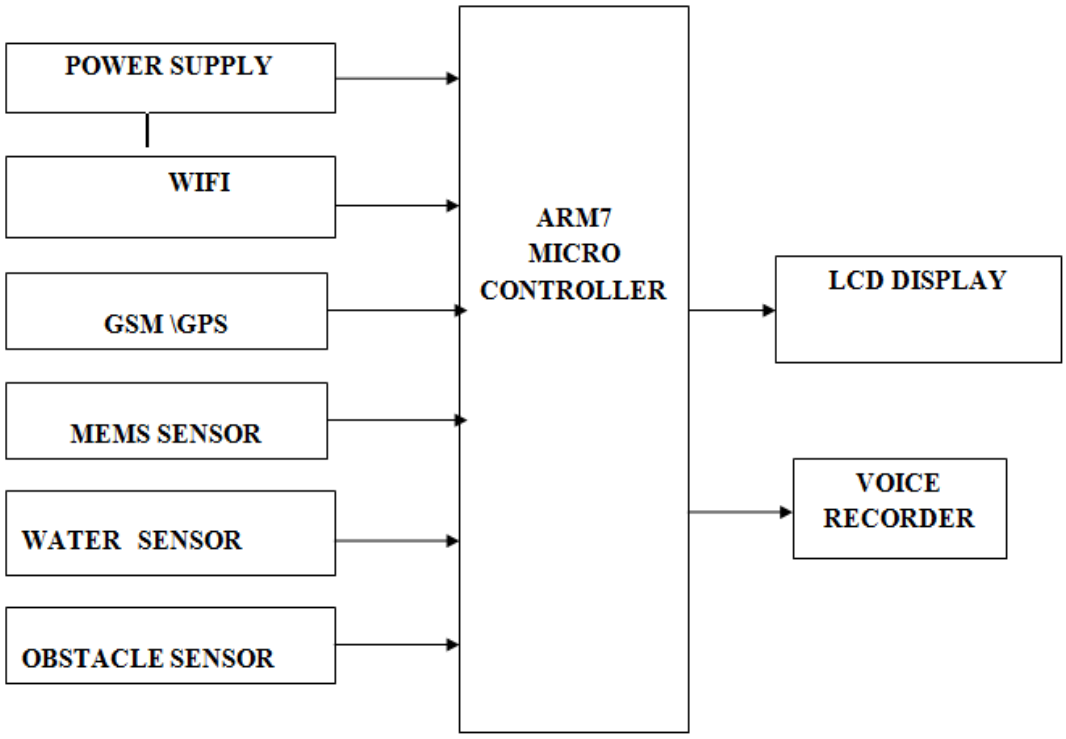


Fig No : 3 System architecture

3.2 Technical architecture

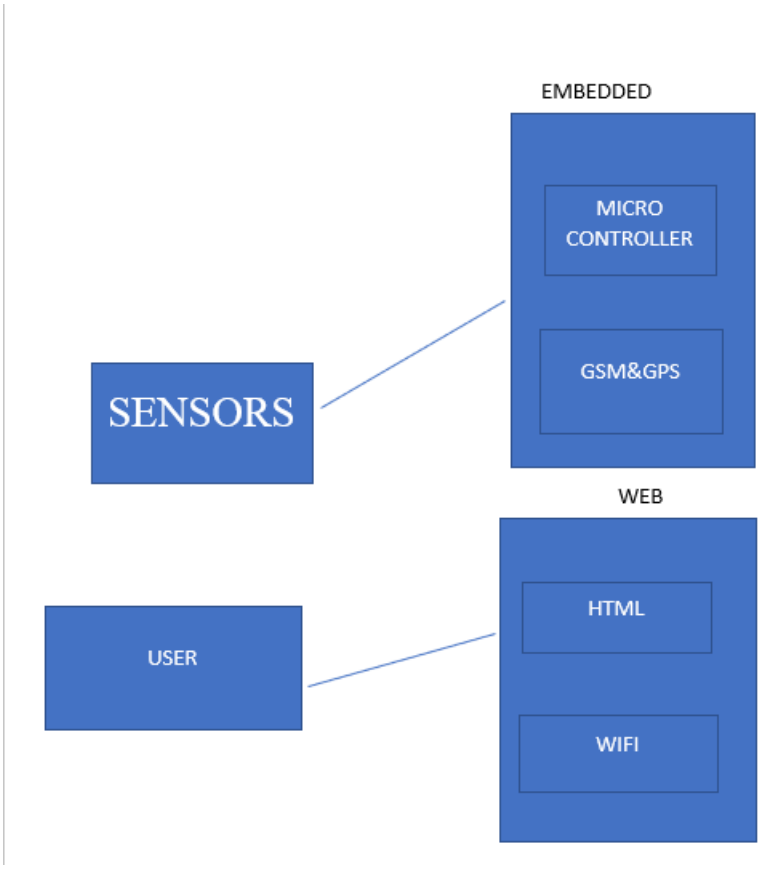


Fig No: 3.1 Technical architecture

5.Design

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

System design is transition from a user-oriented document to programmers or database personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

5.1 UML diagrams

UML Concepts

The UML is language used for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML gives standardized method to draw a system's blueprint.

Goals of UML

The primary goals in the design of UML are:

1. Provide users with a ready to use expensive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes
4. Provide formal basis for understanding the modeling language.

5.1.1 USE CASE DIAGRAM:

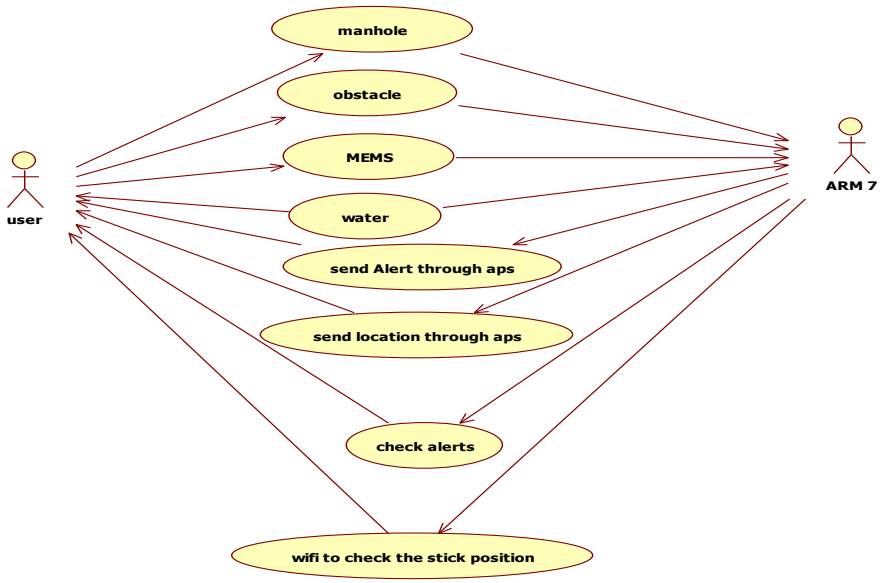


Fig:4.1 Use cse diagram

5.1.2 ACTIVITY DIAGRAM

Activity diagrams represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the state.

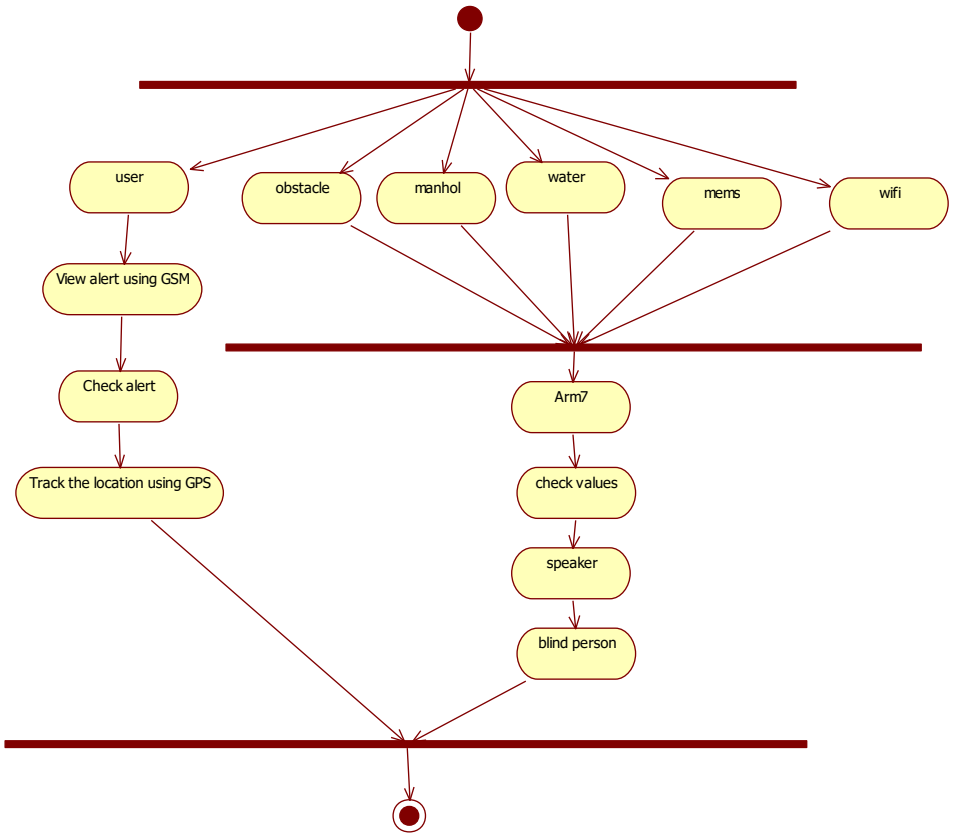


Fig:4.1.1Activity diagram

5.1.3 CLASS DIAGRAM

An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods).

A class is a representation of an object and, in many ways, it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the entire collection of students.

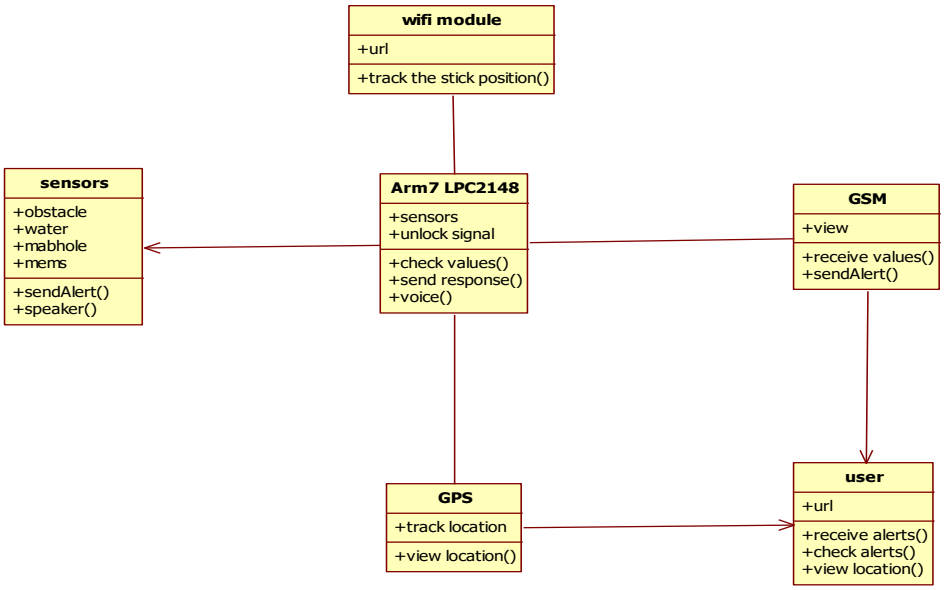
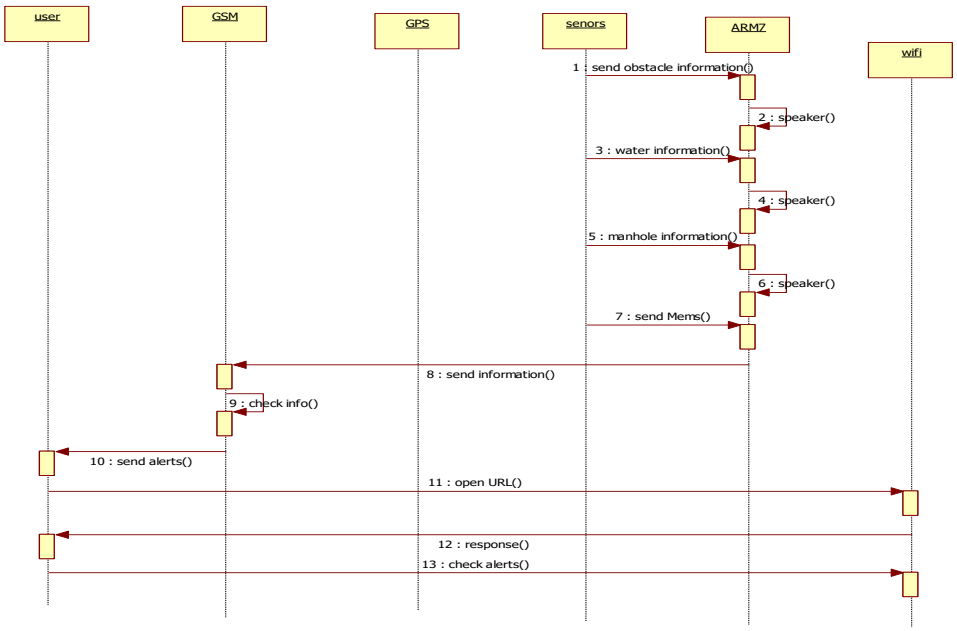


Fig:4.1.2 class diagram

5.1.4 SEQUENCE DIAGRAM

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of vertical dimension(time) and horizontal dimension(objects).



4.1.3 Sequence Diagram

6.MODULES

Server module :

The values which are transferred from the micro controller are storedin the database and we can monitor the values from website.

Microcontroller module:

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes

RPS module:

A regulated power supply is an embedded circuit; it converts unregulated AC into a constant DC. With the help of a rectifier it converts AC supply into DC. Its function is to supply a stable voltage (or less often current), to a circuit or device that must be operated within certain power supply limits. The output from the regulated power supply may be alternating or unidirectional, but is nearly always DC

MEMS Accelerometer Sensor:

Micro-electro-mechanical Systems (MEMS) Technology is one of the most advanced technologies that have been applied in the making of most of the modern devices like video projectors, bi-analysis chips and also car crash airbag sensors

Water detection Sensor:

Water probes are used to detect the oil on the floor.

Manhole and obstacle Detection sensor:

Infrared Obstacle Sensor Module has built-in IR transmitter and IR receiver that sends out IR energy and looks for reflected IR energy to detect presence of any obstacle in front of the sensor module. The module has on board potentiometer that lets user adjust detection range. The sensor has very good and stable response even in ambient light or in complete darkness.

GPS :

GPS navigation device, GPS receiver, or simply GPS is a device that is capable of receiving information from GPS satellites and then to calculate the device's geographical position. Using suitable software, the device may display the position on a map, and it may offer directions. The Global Positioning System (GPS) uses a global navigation satellite system (GNSS) made up of a network of a minimum of 24, but currently 30, satellites placed into orbit.

GSM :

GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation digital cellular networks used by mobile devices such as tablets, first deployed in Finland in December 1991. As of 2014, it has become the global standard for mobile communications – with over 90% market share, operating in over 219 countries and territories.

WIFI:

The **ESPS266** is a low-cost Wi-Fi chip with full TCP/IP stack and MCU(Micro Controller Unit) capability produced by shanghai-based chinese manufacture. The chip first came to the attention of western makers in August 2014 with the ESP-01 module, made by a third-party manufacturer, AI-Thinker. This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

7. PSEUDOCODE

```
#include<LPC214x.H>
#include<string.h>
#define GPIO_Port0s_IODIR    IODIR0
#define GPIO_Port1s_IODIR    IODIR1
#define Set_Port0s          IOSET0
#define Clear_Port0s         IOCLR0
#define Set_Port1s          IOSET1
#define Clear_Port1s         IOCLR1
#define Port0_Set           IOPIN0
#define Port1_Set           IOPIN1
#define Wet_snsr  (1<<12)
#define D_Relay1  (1<<27)
#define D_Relay2       (1<<28)
#define Obstacle_alert1 (1<<16)
#define Obstacle_alert2 (1<<17)
#define Obstacle_alert3 (1<<18)
#define Buzzer          (1<<10)
#defineObstacles  (Obstacle_alert1|Obstacle_alert2|Obstacle_alert3)
#include "LCD.c"
#include "Serial_Uart0.c"
#include "Serial_Uart1.c"
#include "Timmer0.c"
#include "GPS.c"
#include "GSM.c"
#include "I2C.c"
#include "Range_Find.c"
#include "esp.c"
#include "app.c"
int PinStatus_Port(unsigned char ,unsigned int);
void Project_Label(void);
unsigned char x;
unsigned char kmhr_units,kmhr_tens,kmhr_huns;
unsigned char Message_flag=0;
main()
{
GPIO_Port0s_IODIR =~(echo|Wet_snsr);
GPIO_Port0s_IODIR
=(pulser|Obstacle_alert1|Obstacle_alert2|Obstacle_alert3|Buzzer);//|Speed_Alert);
GPIO_Port1s_IODIR = (LCD_Data|RS|EN|D_Relay1|D_Relay2);
Lcd_Init();
I2C_Init();
MEMS_Init();
Init_UART0(9600);
```

```
Init_UART1(9600);
GSM_S900();
ESP_GetReady();
while(1)
                    {
Distance_Measure();
Reading_X_Y_Z_MEMS();
WaterDettect();
          }
}
PinStatus_Port(unsigned char port,unsigned int pin)
{
if(port==0)
{
 x=(Port0_Set& (1<<pin))?1:0;
}
if(port==1)
{
 x=(Port1_Set& (1<<pin))?1:0;
}
 return x;
 }
void Project_Label(void)
{
Lcd_Data_Str(1,1,"  GSM_GPS_MEMS  ");
Lcd_Data_Str(2,1,"Blind AID System");
Delay(150);
Lcd_Data_Chr(0,0,0,LCD_CLEAR);
Lcd_Data_Str(1,1,"Stk:");//5
Lcd_Data_Str(1,9,"Wtr.S:");//15
Lcd_Data_Str(2,1,"Frnt Area:    Cm");//11
}
void Send_AT_Cmd(unsigned char *);
void GSM_Modem_Init(void);
void Check_Response_GPRS_Modem(void);

void Clear_UART_Buffer(void);
void Check_SIM_Registration(unsigned char *);
void Check_Signal_Strength(unsigned char *);
void Check_Client_Request(void);
void Up_Date_WebPage(void);
void Up_Traffic_To_Page(void);
void Check_Traffic(void);
void Check_String(void);
int PinStatus_Port(unsigned char ,unsigned int);
void Get_IPD(void);
unsigned char Channel_Id,Serial_Int_Flag;
unsigned char Connection_ID;
unsigned char j,Command[5],Server_Data[25];
```

```
unsigned char flag1=1,flag2=1,flag3=1;
void ESP_GetReady(void)
{
Disable_UART_Interrupt();
Lcd_Data_Str(1,1,"ESP Initing...");
Delay(200);
Lcd_Data_Str(1,1,"ESP Initing.....");
UART1_TX_Str("AT+RST");
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
Delay(600);
UART1_TX_Str("ATE0");
UART1_TX_Chr(0x0D);

UART1_TX_Chr(0x0A);
Delay(400);
UART1_TX_Str("AT");
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
Delay(400);
Lcd_Data_Str(1,1," ESP Inited Ok ");
Delay(100);
Lcd_Data_Str(1,1,"Set ESP as A.Pt.");
UART1_TX_Str("AT+CWMODE=2");
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
Delay(400);
Lcd_Data_Str(1,1,"ESP as A.Pt. OK ");
Delay(100);
Lcd_Data_Str(1,1,"Set ESP Name&Pwd");
UART1_TX_Str("AT+CWSAP=\"WIFI-B.AID\",\"12345678\",1,3");
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
Delay(600);
Lcd_Data_Str(1,1,"ESP Name&Pwd OK");
Delay(100);
Lcd_Data_Str(1,1,"Set ESP as Servr");
UART1_TX_Str("AT+CIPMUX=1");
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);

Delay(400);
UART1_TX_Str("AT+CIPSERVER=1,80");
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
Delay(400);
Lcd_Data_Str(1,1,"ESP as Servr RDY");
Enable_UART_Interrupt();
}
```

```
void Get_IPD(void)
{
//while(Get_Serial_Data()!='I');
while(UART1_RX_Chr()!='P');
while(UART1_RX_Chr()!='D');
while(UART1_RX_Chr()!=',');
Channel_Id=UART1_RX_Chr();
while(UART1_RX_Chr()!='G');
while(UART1_RX_Chr()!='E');
while(UART1_RX_Chr()!='T');
while(UART1_RX_Chr()!='/');
for(j=0;j<15;j++)
{
Server_Data[j]=UART1_RX_Chr();
if(Server_Data[j]=='/')
{
Server_Data[j]=0x00;
break;
}
}
Lcd_Data_Chr(1,2,16,Channel_Id);
//Lcd_Data_Str(1,14,Server_Data);
Up_Date_WebPage();
        }
void Up_Date_WebPage(void)
{
Disable_UART_Interrupt();
UART1_TX_Str("AT+CIPSEND=");
UART1_TX_Chr(Channel_Id);
UART1_TX_Str(",208");//size of bytes to transfer -->
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
while(UART1_RX_Chr()!='>');
UART1_TX_Str("<!DOCTYPE html>");//15
UART1_TX_Str("<html>");//6
UART1_TX_Str("<meta http-equiv=\"refresh\" content=\"10\">");//40
UART1_TX_Str("<center>");//8
UART1_TX_Str("<h1>IOT Based Blind Stick Assist</h1>");//37
UART1_TX_Str("<br>");//4
UART1_TX_Str("<br>");//4
if(walkflag){
UART1_TX_Str("<h2>Front Way:Clear    ");} //
if(!walkflag){
UART1_TX_Str("<h2>Front Way:Not Clear");}//23
UART1_TX_Str("<br>");//4

if(stickflag){
UART1_TX_Str("<h2>Walking Stick:Stable   ");}//
if(!stickflag){
```

```
UART1_TX_Str("<h2>Walking Stick:Fell Down");}//27
UART1_TX_Str("<br>");//4
if(PinStatus_Port(0,12)==0)
{
UART1_TX_Str("<h2>Water Sensor:Wet");//
}
if(PinStatus_Port(0,12)==1)
{
UART1_TX_Str("<h2>Water Sensor:Dry");//20
}
UART1_TX_Str("</center>");//9
UART1_TX_Str
while(UART1_RX_Chr()!='K');
UART1_TX_Str("AT+CIPCLOSE=");
UART1_TX_Chr(Channel_Id);
UART1_TX_Chr(0x0D);
UART1_TX_Chr(0x0A);
void GSM_S900(void);
void GSM_Modem_Init(void);
void Sim_Registering(void);
void Give_Misscall(unsigned char);
void Registering_Mobile_NO(void);
void Ph_no_Reg_Message_Sending(void);

void Message_Reading(void);
void Chk_Misscall_Message_Send(unsigned char *);
void Sending_Message_For_Misscall(unsigned char *);
void Sending_Message_For_Stored_Misscall(unsigned char *,unsigned char *);
void Message_GPS_Send_To_PhNo(unsigned char *,unsigned char *,unsigned char
*);
void Sending_Message_For_Messg_Rev(unsigned char *);
void Message_Deleting(void);
void Message_String_Compare(void);
void clear_mess_buffer(void);
void GPS_Get_Data_Miscall(void);
void Project_Label(void);
void Sending_GPS_Message_For_Messg_Rev(unsigned char *Mess,unsigned char
*gps);
void GPS_Get_Data(void);
void Get_Phno(void);
void GPS_Init(void);
void GPS_Tack(void);
unsigned char mess[15],mess_phone[16],s[3],misscall_phone[18],sim_reg[4];
unsigned char j,LCD_CLEAR=0x01;
unsigned char gsm_delay=150;
unsigned char Misscall_flag,Message_Read_flag;
unsigned char passg_cnt=0;
unsigned char Loc_Mess[9] ="Car stop";
unsigned char Track[6]    ="Where";
```

```
unsigned char ulock_flag,car_freez=0;
void Message_String_Compare(void)
{

//if(strcmp(misscall_phone,PhNo_string)==0)
//{
//Lcd_Data_Str(2,1,"  Valid Ph No   ");
//Delay(500);
if(strcmp(Track,mess)==0)
{
Lcd_Data_Str(2,1,"Sending Location");
GPS_Tack();
Sending_Message_For_Stored_Misscall("GSM-GPS Vehicle Anti-Theft
Sys\r\nVehicle At:\r\n",gps_data);
clear_mess_buffer();
Delay(100);
}
}
```

GSM Modem Initing functions

```
void GSM_S900(void)
{
GSM_Modem_Init();
Sim_Registering();
Message_Deleting();
Registering_Mobile_NO();
}
//Modem Initing functions
void GSM_Modem_Init(void)
{
Disable_UART0_Interrupt();

Lcd_Data_Str(1,1,"GSM Initng      ");
UART0_TX_Str("AT\r\n");
Delay(gsm_delay);
UART0_TX_Str("AT\r\n");
Lcd_Data_Str(1,11,".");
Delay(gsm_delay);
UART0_TX_Str("AT+CMGF=1\r\n");
Lcd_Data_Str(1,12,".");
Delay(gsm_delay);
UART0_TX_Str("AT+CLIP=1\r\n");
Lcd_Data_Str(1,13,".");
Delay(gsm_delay);
UART0_TX_Str("AT&W\r\n");
//while(UART0_RX_Chr()!='O');
//while(UART0_RX_Chr()!='K');
Delay(gsm_delay);
Lcd_Data_Str(1,14,".");
Delay(gsm_delay);
```

```
UART0_TX_Str("AT+CNMI=1,1,0,0,0\r\n");
UART0_TX_Chr(0x0d);
Lcd_Data_Str(1,15,".");
Delay(gsm_delay);
UART0_TX_Str("AT+CNMI=1,1,0,0,0\r\n");
UART0_TX_Chr(0x0d);
Lcd_Data_Str(1,16,".");
Delay(gsm_delay);

Lcd_Data_Str(1,1,"GSM Inited      ");
}
void Sim_Registering(void)
{
Lcd_Data_Str(1,1,"SIM Chkng       ");
Disable_UART0_Interrupt();
start:
UART0_TX_Str("AT+CREG?");
UART0_TX_Chr(0x0D);
while(UART0_RX_Chr()!=':');
for(j=0;j<4;j++)
{
sim_reg[j]=UART0_RX_Chr();
}
Lcd_Data_Str(2,1,sim_reg);
Delay(100);
if(sim_reg[3]==0x31)
{
Lcd_Data_Str(1,1,"GSM Inited      ");
Lcd_Data_Str(2,1,"SIM  Registered ");
Delay(300);
GPS_Init();
Enable_UART0_Interrupt();
}
else
{
Lcd_Data_Str(2,1,"SIM  Registering");
Delay(700);
```

Phone_No_Registering.....

```
void Registering_Mobile_NO(void)
{
Project_Label();
Delay(300);
 Give_Misscall(Restg);
while(1)
{
if(Reg_Phno_flag==1)
{
//Ph_no_Reg_Message_Sending();
Reg_Phno_flag=0;
```

```
Misscall_flag=0;
Give_Misscall(Restd);
break;
}
}
Project_Label();
Delay(300);
}
void Ph_no_Reg_Message_Sending(void)
{
Disable_UART0_Interrupt();
Lcd_Data_Str(2,1,"message  sending");
UART0_TX_Str ("AT\r\n");
Delay(gsm_delay);
UART0_TX_Str ("AT+CMGS=");
UART0_TX_Str(misscall_phone);
UART0_TX_Chr(0x0D);
UART0_TX_Chr(0x0A);
while(UART0_RX_Chr()!='>');
//UART0_TX_Str ("Your Phone No:");
UART0_TX_Str(misscall_phone);
Delay(50);
UART0_TX_Chr(0x1A);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
Lcd_Data_Str(2,1," message   sent ");
Delay(300);
Lcd_Data_Chr(0,0,0,LCD_CLEAR);
Enable_UART0_Interrupt();
}
void Give_Misscall(unsigned char ph)
{
if(!ph)
{
Lcd_Data_Str(1,1,"Give Miscal     ");
Lcd_Data_Str(2,1,"To Reg Phone No ");
}
if(ph)
{
Lcd_Data_Str(1,1,"Ur Ph No        ");

Lcd_Data_Str(2,1,"Reg Successfully");
Delay(300);
}
}
 Chk_for_MissCall--->Cut_Call--->Send_Messg
void Get_Phno(void)
{
Disable_UART0_Interrupt();
```

```
Lcd_Data_Str(1,1,"               ");
Lcd_Data_Str(1,1,"Misscall       ");
Delay(500);
while(UART0_RX_Chr()!='C');
while(UART0_RX_Chr()!='L');
while(UART0_RX_Chr()!='I');
while(UART0_RX_Chr()!='P');
while(UART0_RX_Chr()!=':');
while(UART0_RX_Chr()!=' ');
for(j=0;j<15;j++)
{
misscall_phone[j]=UART0_RX_Chr();
}
UART0_TX_Str("ATH");
UART0_TX_Chr(0X0D);
UART0_TX_Chr(0X0A);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');

Lcd_Data_Str(1,1,misscall_phone);
Delay(200);
}
void Chk_Misscall_Message_Send(unsigned char *Mess)
{
if(Misscall_flag==1)
{
Misscall_flag=0;
Disable_UART0_Interrupt();
Lcd_Data_Str(2,1,"message  sending");
UART0_TX_Str ("AT\r\n");
Delay(gsm_delay);
UART0_TX_Str ("AT+CMGS=");
UART0_TX_Str(misscall_phone);
UART0_TX_Chr(0x0D);
UART0_TX_Chr(0x0A);
while(UART0_RX_Chr()!='>');
UART0_TX_Str(Mess);
Delay(gsm_delay);
UART0_TX_Chr(0x1a);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
Lcd_Data_Str(2,1," message   sent ");
Delay(300);
Project_Label();
Enable_UART0_Interrupt();
}
}
void Chk_Misscall_GPS_Message_Send(unsigned char *Mess,unsigned char
*gpsvalue)
```

```
{
if(Misscall_flag==1)
{
Misscall_flag=0;
Disable_UART0_Interrupt();
GPS_Get_Data_Miscall();
Delay(gsm_delay);
Lcd_Data_Str(2,1,"message  sending");
UART0_TX_Str ("AT\r\n");
Delay(gsm_delay);
UART0_TX_Str ("AT+CMGS=");
UART0_TX_Str(misscall_phone);
UART0_TX_Chr(0x0D);
UART0_TX_Chr(0x0A);
while(UART0_RX_Chr()!='>');
UART0_TX_Str(Mess);
UART0_TX_Str("http://maps.google.co.in/maps?q=");
UART0_TX_Str(gpsvalue);
UART0_TX_Str("\r\nTotal Passengers Count:");
UART0_TX_Chr((passg_cnt/10)+48);
UART0_TX_Chr((passg_cnt%10)+48);
Delay(gsm_delay);
UART0_TX_Chr(0x1a);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
Lcd_Data_Str(2,1," message   sent ");
Delay(300);
Project_Label();
Enable_UART0_Interrupt();
}
}
```

Alert_Sending_Message_For_Received_Message

```
void Sending_GPS_Message_For_Messg_Rev(unsigned char *Mess,unsigned char
*gps)
{
Disable_UART0_Interrupt();
Lcd_Data_Str(2,1,"message  sending");
UART0_TX_Str ("AT\r\n");
Delay(gsm_delay);
UART0_TX_Str ("AT+CMGS=");
UART0_TX_Str(mess_phone);
UART0_TX_Chr(0x0D);
UART0_TX_Chr(0x0A);
while(UART0_RX_Chr()!='>');
UART0_TX_Str(Mess);
UART0_TX_Str("http://maps.google.co.in/maps?q=");
UART0_TX_Str(gps);
Delay(gsm_delay);
UART0_TX_Chr(0x1a);
```

```
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
Lcd_Data_Str(2,1," message   sent ");

Delay(1000);
Project_Label();
Enable_UART0_Interrupt();
}
void Sending_Message_For_Messg_Rev(unsigned char *Mess)
{
Disable_UART0_Interrupt();
Lcd_Data_Str(2,1,"message  sending");
UART0_TX_Str ("AT\r\n");
Delay(gsm_delay);
UART0_TX_Str ("AT+CMGS=");
UART0_TX_Str(mess_phone);
UART0_TX_Chr(0x0D);
UART0_TX_Chr(0x0A);
UART0_TX_Str(Mess);
while(UART0_RX_Chr()!='>');
//UART0_TX_Str(Gpslink);
Delay(gsm_delay);
UART0_TX_Chr(0x1a);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
Lcd_Data_Str(2,1," message   sent ");
Delay(300);
Enable_UART0_Interrupt();
}
```

Message Deleting

```
void Message_Deleting(void)
{
Lcd_Data_Chr(0,0,0,LCD_CLEAR);
Lcd_Data_Str(2,1,"Mess over load..");
Lcd_Data_Str(2,1,"Deleting.....");
Disable_UART0_Interrupt();
for(j=0X31;j<=0X39;j++)
{
UART0_TX_Str("AT+CMGD=");
UART0_TX_Chr(j);
UART0_TX_Chr(0X0D);
UART0_TX_Chr(0X0A);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
if(j==0x39)
{
UART0_TX_Str("AT+CMGD=");
```

```
UART0_TX_Chr(0x31);
UART0_TX_Chr(0x30);
UART0_TX_Chr(0X0D);
UART0_TX_Chr(0X0A);
while(UART0_RX_Chr()!='O');
while(UART0_RX_Chr()!='K');
Lcd_Data_Str(2,1,"    Deleted     ");
Delay(300);

Enable_UART0_Interrupt();
void clear_mess_buffer(void)
{
for(j=0;j<16;j++)
{
mess[j]=0x00;
}
}
void Get_ADC_Data(void);
void ADC0_Channel_3(void);
void ADC0_Channel_2(void);
unsigned char WtrMflag=1;
void WaterDettect(void)
{
//Lcd_Data_Str(1,9,"Wtr.S:");//15
if(PinStatus_Port(0,12)==1)
{
Lcd_Data_Str(1,15,"Dy");//15
 WtrMflag=1;
}
else
{
Lcd_Data_Str(1,15,"Wt");//15
if(WtrMflag)
{
WtrMflag=0;
GPS_Get_Data_Miscall();


 Sending_Message_For_Stored_Misscall("Blind Aid System\r\n Wet Detected
At\r\n",gps_data);
 Clear_Port0s=(Obstacle_alert3);
  Delay(150);
  Set_Port0s=(Obstacle_alert3);
  Delay(1000);
}
}
}
#define echo   (1<<6)
#define pulser (1<<7)
```

```
void Pulse(void);
void Pulse_Delay(unsigned int);
void Distance_Measure(void);
void Lcd_Data_Display(void);
void Find_Distance(void);
void Hex_Ascii(int);
void Lcd_Data_Display(void);
void Stop_Timmer0(void);
sssint PinStatus_Port(unsigned char ,unsigned int);
unsigned int count;
unsigned int value,msb_value,lsb_value;
unsigned long time;
float time_us,Distance_cm,time_ms;

int d1,d2,d3,d4;
unsigned char  Timer_flag,walkflag=1,DistMflag=1;
unsigned int TIME;

//int PinStatus_Port(unsigned char ,unsigned char);
void Distance_Measure(void)
{
Pulse();
while(PinStatus_Port(0,6)!=1);
Timmer0_Registers_Init();
while(PinStatus_Port(0,6)!=0&&Timer_flag!=1);
{
if(Timer_flag==1)
{
Stop_Timmer0();
Timer_flag=0;
}
if(Timer_flag==0)
{
Find_Distance();
 }
}
}
void Find_Distance(void)
{
TIME=T0TC;

Stop_Timmer0();
TIME=TIME/10;
Distance_cm=TIME/58;
Hex_Ascii(Distance_cm);
Lcd_Data_Display();    //slow speed
if(Distance_cm<=10)
{
Lcd_Data_Str(2,11,"Okay  ");//11
```

```
walkflag=1;
DistMflag=1;
}
else
{
walkflag=0;
Lcd_Data_Str(2,11,"Not Ok");
if(DistMflag)
{
DistMflag=0;
GPS_Get_Data_Miscall();
Sending_Message_For_Stored_Misscall("Blind Aid System\r\nMan Hole or Hole
Detected At\r\n",gps_data);
Clear_Port0s=(Obstacle_alert2);
Delay(150);
Set_Port0s=(Obstacle_alert2);
Delay(1000);
}
```

8. INPUT SCREENSHOTS

For GSM Initiating :

Fig No 5: For GSM Initiating

For Mem Initiating:

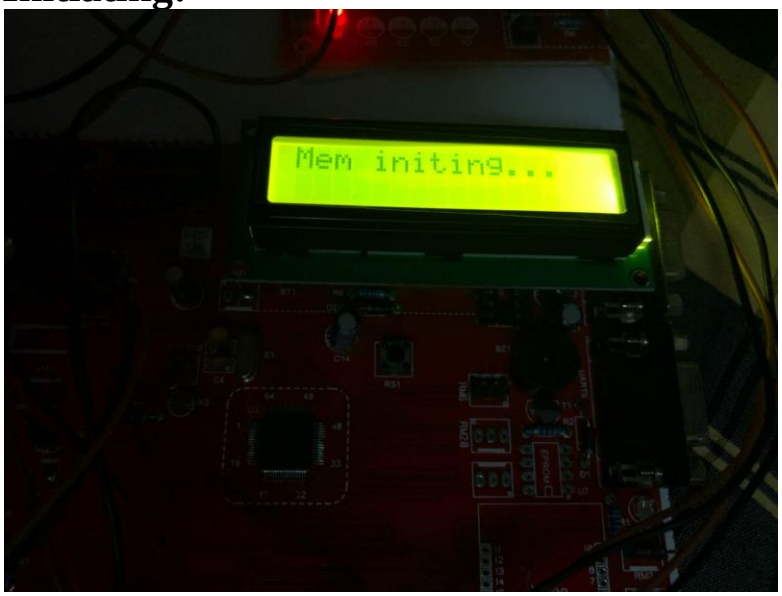


Fig No 5.1: For Mem Initiating

For Deleting:

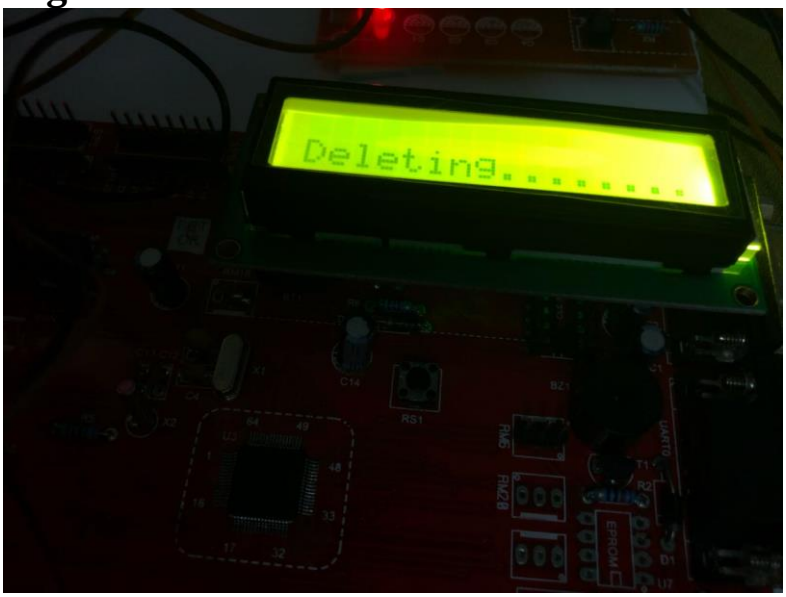


Fig No 5.1: For Deleting

For SIM Checking:

Fig No 5.2: For SIM Checking

For Phone Number Registration

Fig No 5.3 For Phone Number Registration

For Missed Call

Fig No : 5.4 For Missed Call

For To Register Phone No:

Fig No : 5.5 For To Register Phone No

For Front Area:

Fig No : 5.6 For Front Area

Web Page:

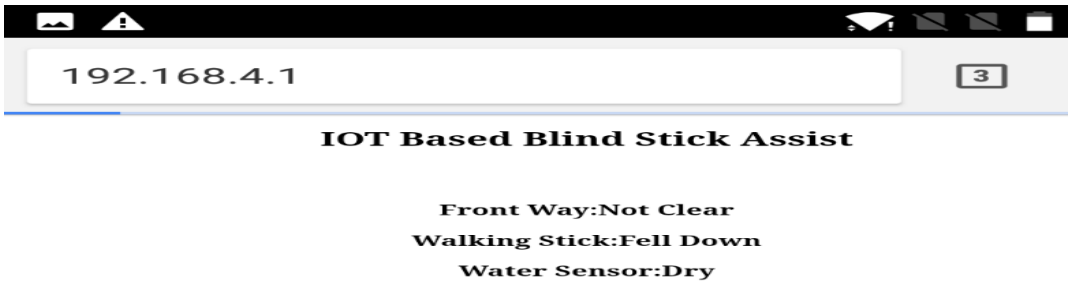


fig No : 5.6 Web page

9. CONCLUSION

The project "IOT **assistance based smart walking stick with GPS and GSM for blind old peoples"** has been successfully designed and tested. It has been developed by integrating features of all the hardware components used. 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 and with the help of growing technology the project has been successfully implemented.

10.BIBILOGRAPHY

1. **WWW.MITEL.DATABOOK.COM**
2. **WWW.ATMEL.DATABOOK.COM**
3. **WWW.FRANKLIN.COM**
4. **WWW.KEIL.COM**

10. REFERENCES

1. "The 8051 Microcontroller Architecture, Programming & Applications" By Kenneth J Ayal.
2. "The 8051 Microcontroller & Embedded Systems" by Mohammed Ali Mazidi and Janice Gillispie Mazidi
3. "Power Electronics" by M D Singh and K B Khanchandan
4. "Linear Integrated Circuits" by D Roy Choudary & Shail Jain
5. "Electrical Machines" by S K Bhattacharya
6. "Electrical Machines II" by B L Thereja
7. www.8051freeprojectsinfo.com