ABSTRACT

This project deals with the concept which can detect accidents without any human assistance. Detection of accidents is done automatically by using a simple setup that will be embedded in the vehicle. Once the vehicle met with an accident the accident detection setup will sense the accident and immediately sends the location of the accident to a person. After receiving the coordinates to the accident spot the person will be rushed to the same. In urban areas accidents are most common phenomena where many of such accidents can be taken care easily but some accidents occur during night time when the visibility is quite low, during such cases it will be difficult for an driver to identify the accident spot with the help of phone calls made by the citizens. If the driving force knows the precise spot of the accident the time period between the spot and the hospital is going to be significantly reduced. The main objective of this project is to help and reduce the time factor in case of accidents. There are many cases where an accident occurs during the night and the person met with the accident is unconscious then it would take hours for someone to find out and inform the authorities about it. So saving such precious time will indeed save lives. In connection with this concept, an experimental setup is constructed that can detect accidents automatically without any human help. After the accident detection, the same setup will send accident coordinates to the person to help and find the location easily

CHAPTER - 1

EMBEDDED SYSTEM

1.1Introduction :

An embedded system is a computer system with a dedicated function within a larger, often with, real-time mechanical or electrical system constraints. It is incorporated as part of a complete device that often includes hardware and mechanical parts. Integrated systems today control many commonly used devices. 92% of all microprocessors are produced as embedded system components.

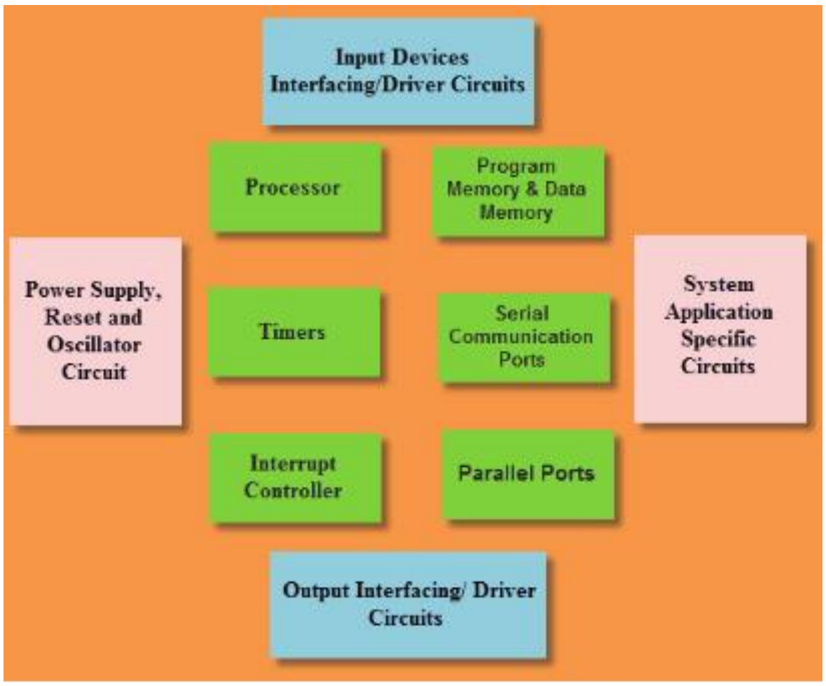

Input Devices Interfacing/Driver Circuits
Processor
Program Memory & Data Memory
Power Supply, Reset and Oscillator Circuit
Timers
Serial Communication Ports
System Application Specific Circuits
Interrupt Controller
Parallel Ports
Output Interfacing/ Driver Circuits

Fig 1.1: Embedded Systems Design

Examples of typically incorporated computer properties, compared to general-purpose counterparts, are low power consumption, small size, varying performance and low-cost unit cost. This is at the expense of limited processing resources, which makes them much harder to program and interact. However, by building intelligence mechanisms at the top of the hardware, taking advantage of any existing sensors and the existence of a built-in network can best manage the available network and unit resources.

For example, intelligent techniques can be designed to manage the energy consumption of embedded systems. Modern embedded systems often incorporate microcontrollers (i.e. CPUs with memory or peripheral interfaces) but common microprocessors are also commonly used (using external memory chips and peripheral circuitry interfaces), especially in more complex systems. In any case, the processor or processors can be used typed ranging from general-purpose specialized in some kind of calculations, or even designed for the application. A common standard of dedicated processors is the digital signal processor (DSP).

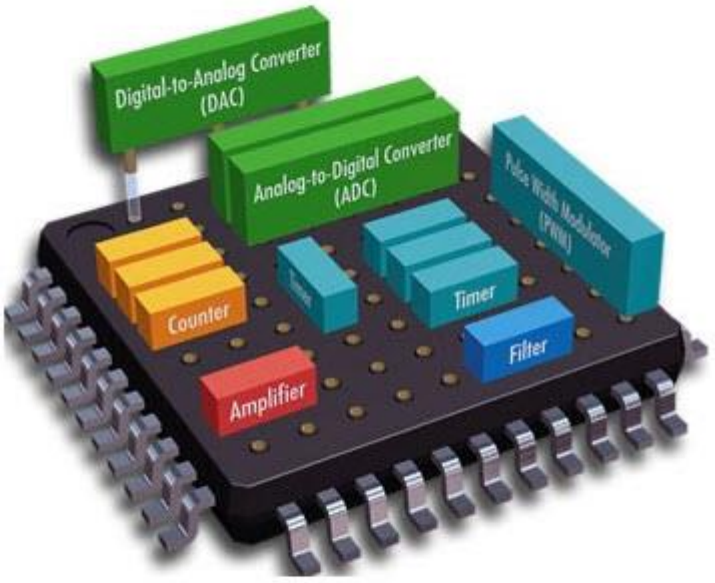

Digital-to-Analog Converter (DAC)
Analog-to-Digital Converter (ADC)
Pulse Width Modulator (PWM)
Counter
Timer
Timer
Filter
Amplifier

Fig 1.2: Embedded System Hardware

Since the integrated system is dedicated to specific tasks, designers can optimize in order to reduce the size and cost of the product and increase reliability and performance. Some embedded systems are serially produced, benefiting from economies of scale.

Integrated systems range from portable devices, such as digital clocks and MP3 players, to large stationary systems such as traffic lights, factory controllers and large complex systems such as hybrid vehicles, magnetic resonances, and avionics. The complexity varies from low to high, with a single microcontroller chip, with the highest number of units, peripherals, and networks mounted within a large frame or enclosure.

1.2 History:

One of the very first recognizably modern embedded systems was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory. At the project's inception, the Apollo guidance computer was considered the riskiest item in the Apollo project as it employed the then newly developed monolithic integrated circuits to reduce the size and weight. An early mass-produced embedded system was the Automatics D-17 guidance computer for the Minuteman missile, released in 1961. When the Minuteman II went into production in 1966, the D-17 was replaced with a new computer that was the first high-volume use of integrated circuits.

Since these early applications in the 1960s, embedded systems have come down in price and there has been a dramatic rise in processing power and functionality. An early microprocessor for example, the Intel 4004, was designed for calculators and other small systems but still required external memory and support chips. In 1978 National Engineering Manufacturers Association released a "standard" for programmable microcontrollers, including almost any computer-based controllers, such as single board computers, numerical, and event-based controllers.

As the cost of microprocessors and microcontrollers fell it became feasible to replace expensive knob-based analog components such as potentiometers and variable capacitors with up/down buttons or knobs read out by a microprocessor even in consumer products. By the early 1980s, memory, input and output system components had been integrated into the same chip as the processor forming a microcontroller. Microcontrollers find applications where a general-purpose computer would be too costly.

A comparatively low-cost microcontroller may be programmed to fulfill the same role as a large number of separate components. Although in this context an embedded system is usually more complex than a traditional solution, most of the complexity is contained within the microcontroller itself.

1.3 Embedded System Classification:

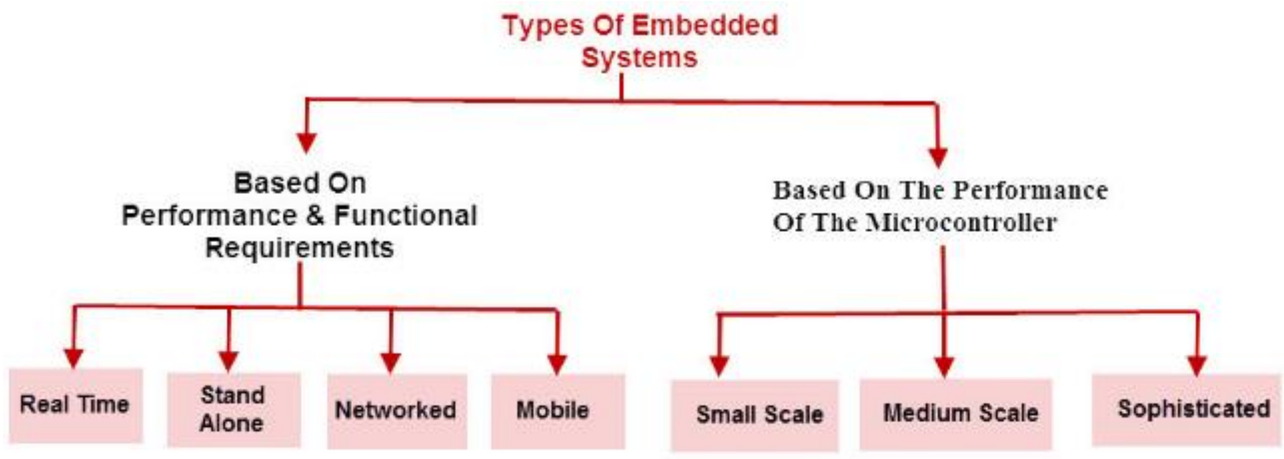

Types Of Embedded Systems
Based On Performance & Functional Requirements
Based On The Performance Of The Microcontroller
Real Time
Stand Alone
Networked
Mobile
Small Scale
Medium Scale
Sophisticated

Fig 1.3: Classification of Embedded Systems

Embedded systems are primarily classified into different types based on complexity of hardware & software and microcontroller (8 or 16 or 32-bit). Thus, based on the performance of the microcontroller, embedded systems are classified into three types such as:

- Small scale embedded systems
- Medium scale embedded systems
- Sophisticated embedded systems

Further, based on performance and functional requirements of the system embedded system classified into four types such as:

- Real time embedded systems
- Standalone embedded systems
- Networked embedded systems
- Mobile embedded systems

1.4 Applications:

Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical, commercial and military applications.

Telecommunications systems employ numerous embedded systems from telephone switches for the network to cell phones at the end user. Computer networking uses dedicated routers and network bridges to route data.

Fig1.4: Real-life examples of Embedded System

Consumer electronics include MP3 players, mobile phones, videogame consoles, digital cameras, GPS receivers, and printers. Household appliances, such as microwave ovens, washing machines and dishwashers, include embedded systems to provide flexibility, efficiency and features. Advanced HVAC systems use networked thermostats to more accurately and efficiently control temperature that can change by time of day and season. Home automation uses wired- and wireless-networking that can be used to control lights, climate, security, audio/visual, surveillance, etc., all of which use embedded devices for sensing and controlling.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors brushless DC motors, induction motors and DC motors use electric/electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.

Medical equipment uses embedded systems for vital signs monitoring, electronic stethoscopes for amplifying sounds, and various medical imaging (PET, SPECT, CT, and MRI) for non-invasive internal inspections. Embedded systems within medical equipment are often powered by industrial computers.

Embedded systems are used in transportation, fire safety, safety and security, medical applications and life critical systems, as these systems can be isolated from hacking and thus, be more reliable. For fire safety, the systems can be designed to have greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.

1.5 Characteristics:

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

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

1.6 User interface:

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

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

Some systems provide user interface remotely with the help of a serial (e.g. RS-232, USB, I^2C, etc.) or network (e.g. Ethernet) connection. This approach gives several advantages: extends the capabilities of embedded system, avoids the cost of a display, simplifies BSP and allows one to build a rich user interface on the PC. A good example of this is the combination of an embedded web server running on an embedded device (such as an IP camera) or a network router. The user interface is displayed in a web browser on a PC connected to the device, therefore needing no software to be installed.

1.7 Processors in Embedded Systems:

Embedded processors can be broken into two broad categories. Ordinary microprocessors (μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have on-chip peripherals, thus reducing power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architecture comes in a large number of different variants and shapes, many of which are also manufactured by several different companies.

1.8 Peripherals:

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

- Serial Communication Interfaces (SCI): RS-232, RS-422, RS-485, etc.
- Synchronous Serial Communication Interface: I2C, SPI, SSC and ESSI (Enhanced Synchronous Serial Interface)
- Universal Serial Bus (USB)
- Multi Media Cards (SD cards, Compact Flash, etc.)
- Networks: Ethernet, Lon Works, etc.
- Fieldbuses: CAN-Bus, LIN-Bus, PROFIBUS, etc.
- Timers: PLL(s), Capture/Compare and Time Processing Units
- Discrete IO: aka General Purpose Input/Output (GPIO)
- Analog to Digital/Digital to Analog (ADC/DAC)
- Debugging: JTAG, ISP, ICSP, BDM Port, BITP, and DB9 ports.

CHAPTER – 2

ARDUINO UNO

2.1 Introduction:

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicating with software running on your computer. The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free. The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

2.2 What is Arduino ?

Arduino is an open source electronics platform accompanied with a hardware and software to design, develop and test complex electronics prototypes and products. The hardware consists of a microcontroller with other electronic components which can be programmed using the software to do almost any task. The simplicity of the Arduino language makes it very easy for almost everyone who has an interest in electronics to write programs without the understanding of complex algorithms or codes.

Arduino is intended for an artist, tinker, designer or anyone, interested in playing with electronics without the knowhow of complex electronics and programming skills. Arduino is an excellent designed open source platform. It has specially designed boards which can be programmed using the Arduino Programming Language (APL).
The open source nature of Arduino has been the main reason for its rapid horizontal growth. Since it is an Open Source project, all the files related to hardware and software is available for personal or commercial use. The development cost of the hardware is very small as against the costly similar proprietary products by the industrial giants.

2.3 Concept of Arduino :

The root of Arduino goes deep down to the development of Processing Language by MIT researchers. Processing language is an open source language designed to introduce the software development environment for the artistic people without the need of deep knowledge of programming of algorithms. Processing is based on java.

In early year of 21st century, designing an electronics gadget was nearly impossible for a common man. The requirement of specific skill set and hefty prices of software and hardware created a full stop in the path of their creativity.

In year 2003 Hernando Barragan, a programmer developed an open source electronics development platform with software IDE, where anyone with a small knowledge in electronics and programming could use his project to give wings to their creativity. His focus was to reduce the burden of complexity in designing electronics hardware and software. The project was named as Wiring. The software IDE of the Wiring used processing language to write the codes.

As the program written in C\C++ is named as *Project*, in the same way the code written in Wiring (even in Processing and Arduino) is termed as *Sketch*. The name sketch gives a familiar look for an artist.

Wiring has predefined libraries to make the programming language easy. Arduino uses these libraries. The predefined libraries are written in C and C++. One can even write his software in C\C++ and use them on *wiring* boards. The difference between writing a program in C/C++ and Wiring is that the Wiring *Application Programmable Interface* (API) has simplified programming style and the user doesn't require detailed knowledge of the concepts like classes, objects, pointers, etc. While sketching hardware you need to call the predefined functions and rest will be handled by the Wiring software.

2.4 History:

Wiring is the predecessor of Arduino. Arduino was developed in lvrea, Italy by Massimo Banzi and David Cuartielles in year 2005. The Project was named after Arduin of lvrea (King of Italy). The project Arduino uses the Wiring language. The concept of Wiring Language was created by Hernando Barragan, and under his supervision Massimo Banzi and David Cuartielles developed the Project Arduino.

2.5 Open source license:

Arduino is an open source project which is probably the root cause reason for its popularity. Arduino hardware design is an Open Source Hardware, distributed under ***Creative Common*** *Attribution Share-Alike license*. Creative Common, a non-profitable organization has released several copyleft-licenses as free of charge, so that the creativity/ knowledge can be shared to the rest of the world while having the copyright to the authorized person. The originally designed files, like layout and schematics of Arduino products are available as Eagle CAD files.

The source code for its IDE and libraries are also available and released under GUN General Public License (known as GPL). The GPL is the first copyleft license for general use. The license is granted for the software to ensure the copyleft freedom.

2.6 ATMEL ATMEGA328P:

The ATmega328P chip is used in this project as the microcontroller. The significance of the first two digits is to stipulate that the AVR core consists of variety of instruction set with 32 general purpose working registers which are connected directly to the Arithmetic Logical Unit (ALU), tolerating two independent registers to be retrieved in one single instruction executed in one clock cycle. The subsequent architecture is more programmable efficient while attaining data transfer rates up to ten times quicker than other CISC microcontrollers. The last digit is to indicate the 8 bit bi-directional port. It is certainly the head of the system which is controlling the various modules. The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996. The AVR was one of the first microcontroller families to use on-chip flash memory for program storage, as opposed to one-time programmable ROM, EPROM, or EEPROM used by other microcontrollers at the time.

The Arduino hardware was very skillfully designed to reduce the complexities arising in the circuitry. It has an In System Programmer (ISP), which allows users to transfer the software inside the microcontroller without removing it from the circuit. The basic model of an Arduino board consists of an 8-bit AVR microcontroller along with some other necessary components like a 5 volt linear regulator IC, a 16 MHZ crystal, ceramic resonator, output connectors, direct adaptor input, etc.

The IO ports on boards are positioned in a way that it can be easily attached with the interchangeable add-on modules, known as shields. Shields are daughter boards that can be externally attached/ plugged with the Arduino boards to extent the board's capabilities. For example an xbee shield can be attached with the Arduino board to establish a wireless communication. A motor control shield can be attached on the top of Arduino board to run the motors or to provide an ease to control the speed of motors. The Arduino Board can easily interface with external sensors, circuits or other peripherals.

The high-performance Atmel Pico Power 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed (Atmel Corporation).

2.7 Arduino UNO:

To program the ATmega328P Microcontroller a Serial communicator is required. Serial communication is most widespread interface between microcontroller and computer. UART is one of the serial interfaces which are widely used. A Universal Asynchronous Receiver/Transmitter (UART) is a piece of computer hardware that translates data between parallel and serial forms. Classically, most serial interface from microcontroller to computer is done through serial port (DB9). However, since computer serial port used RS232 protocol and microcontroller used TTL UART, a level shifter is needed between these interfaces. There are several level shifters available in the market, some of which supports USB plug and play. Arduino UNO is an alternative to this solution, the internal board of Arduino consists of all the necessary ICs for communication. It is also build compact into a PCB which has connectors for fast and easy prototyping.

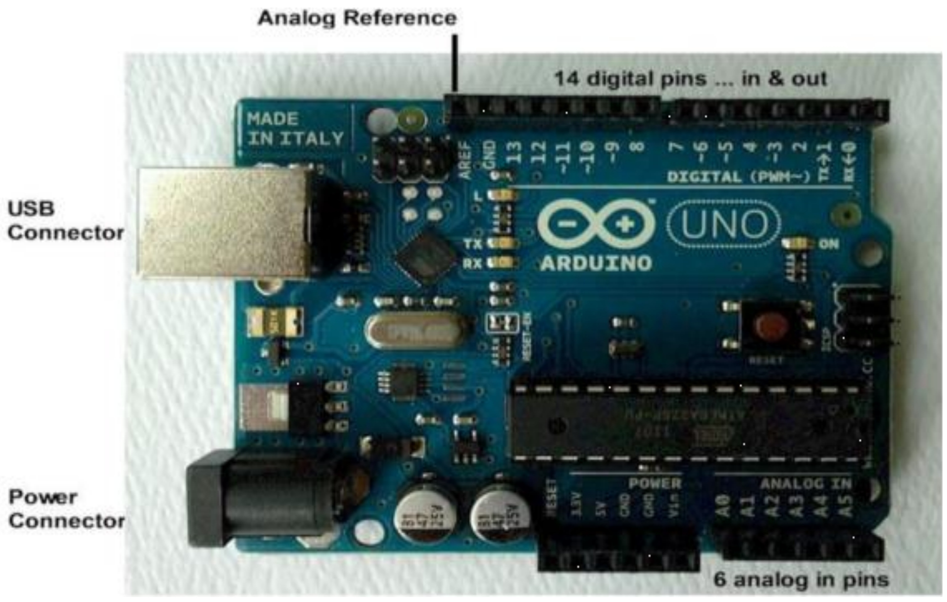


Fig 2.1: Arduino UNO Board

2.8 Summary:

- ✓ Microcontroller ATmega328
- ✓ Operating Voltage 5V
- ✓ Input Voltage (recommended) 7-12V
- ✓ Input Voltage (limits) 6-20V
- ✓ Digital I/O Pins 14 (of which 6 provide PWM output)
- ✓ Analog Input Pins 6
- ✓ DC Current per I/O Pin 40 mA

- ✓ DC Current for 3.3V Pin 50 mA
- ✓ Flash Memory 32 KB (ATmega328) of which 0.5 KB used by bootloader
- ✓ SRAM 2 KB (ATmega328)
- ✓ EEPROM 1 KB (ATmega328)
- ✓ Clock Speed 16 MHz

2.9 Pin configuration:

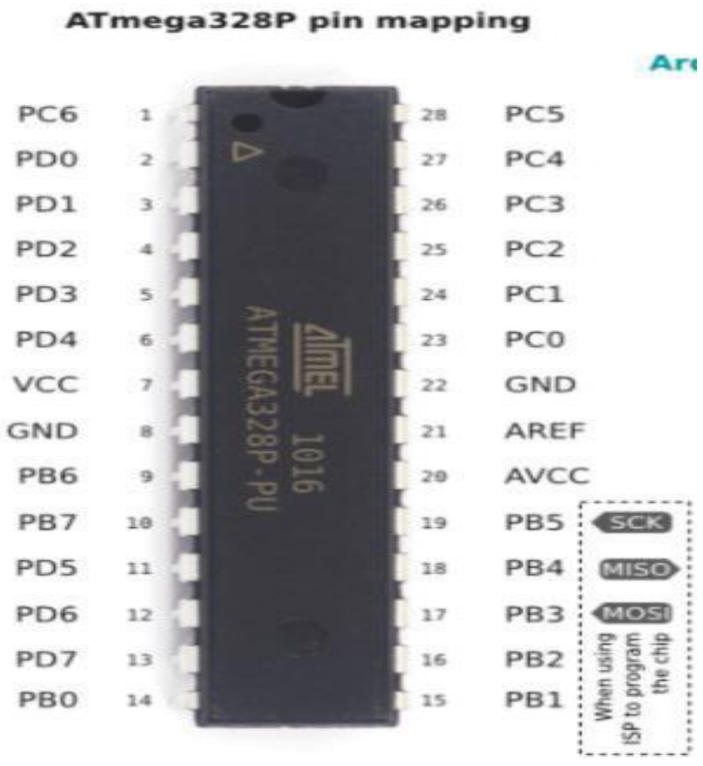


Fig 2.2: ATMEGA328P-PU

- ➢ The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
- ➢ External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

- The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

THE POWER PINS ARE AS FOLLOWS:

- **V$_{IN}$**: The input voltage to the Arduino board when it's using an external power source (as Opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **5V:** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board.
- **3V:** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND**: Ground pins.
- **IOREF:** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

2.10 Memory:

The ATmega328 has 32 KB (with 0.5 KB used for the boot loader). It also has 2 KB of SRAM and 1 KB of EEPROM.

2.11 Input and output:

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

- **Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.

- These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts:** 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM:** 3, 5, 6, 9, 10, and 11**.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI:** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built- in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference() function. Additionally, some pins have specialized functionality:

- **TWI:** A4 or SDA pin and A5 or SCL pin**.** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

2.12 Communication:

Microcontrollers depend on a host computer for developing and compiling programs. The software used on the host computer is known as an integrated development environment, or IDE. For the Arduino, the development environment is based on the open source Processing platform (www.processing.org) which is described by its creators as a “programming language and environment for people who want to program images, animation, and interactions.

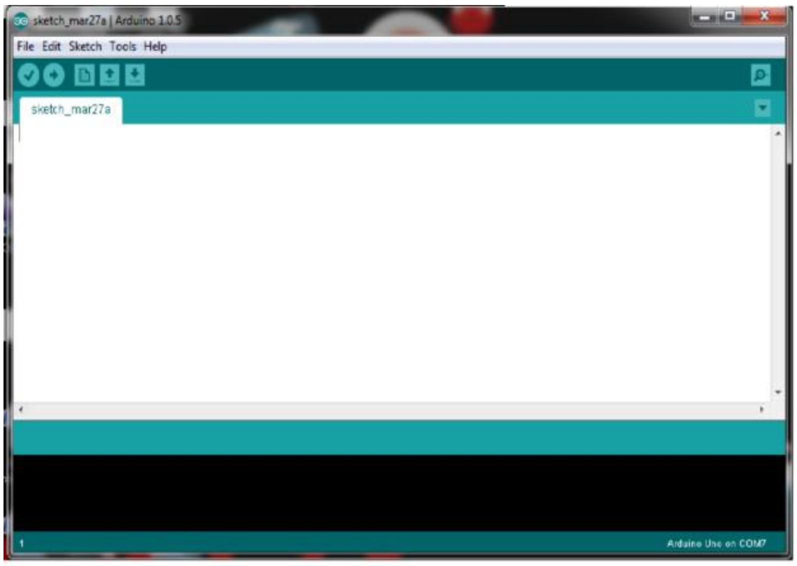
The Arduino Uno has a number of facilities for communicating with a computer, another Arduino or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a info file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

2.13 ARDUINO IDE:

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It includes a code editor which is capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch”.

FOLLOWING ARE THE STEPS INVOLVED:

1. OPEN ARDUINO IDE AS SHOWN BELOW

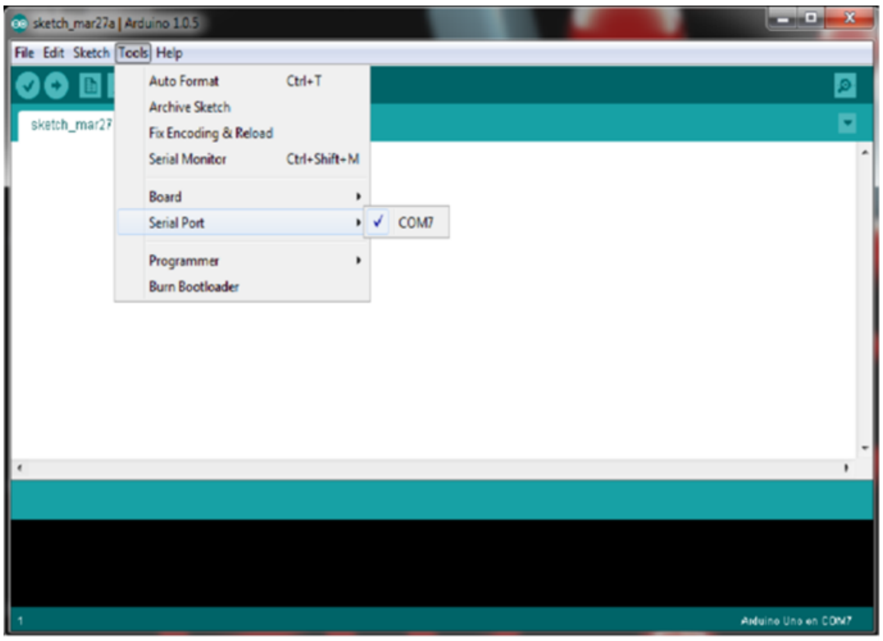


Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much efficient. Users only need define two functions to make a runnable cyclic executive program:

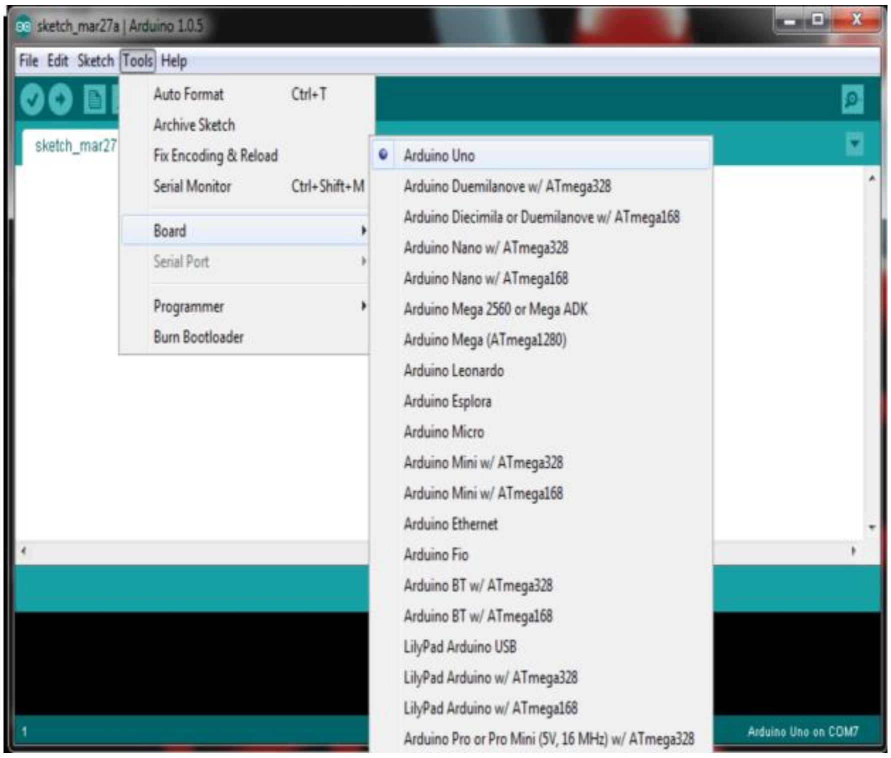
setup(): a function run once at the start of a program that can initialize settings

loop(): a function called repeatedly until the board powers off

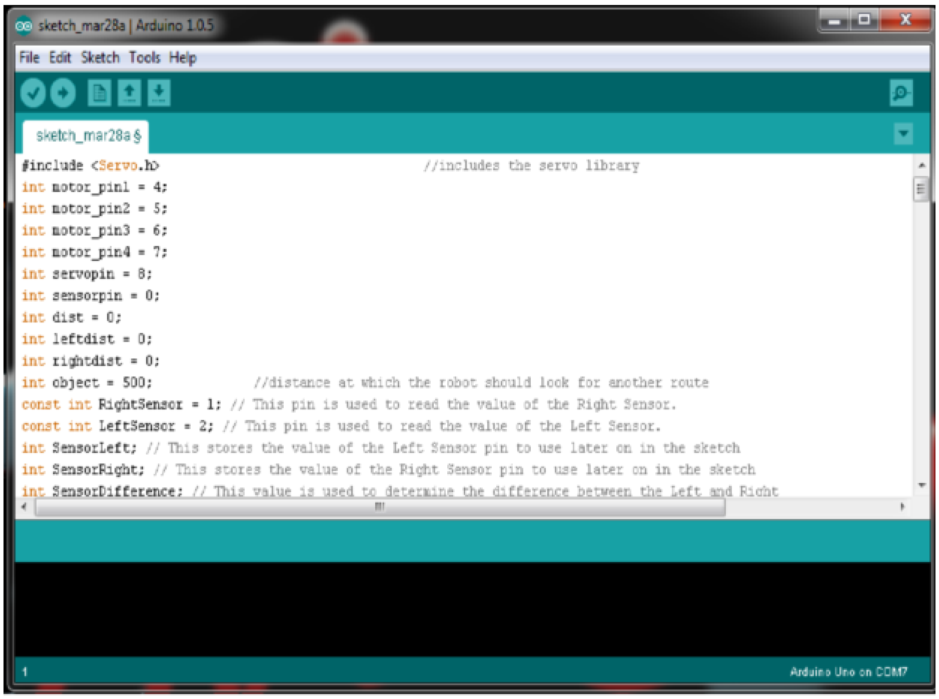
2. SELECT THE COM PORT FROM TOOLS



3. SELECT THE REQUIRED ARDUINO BOARD FROM TOOLS

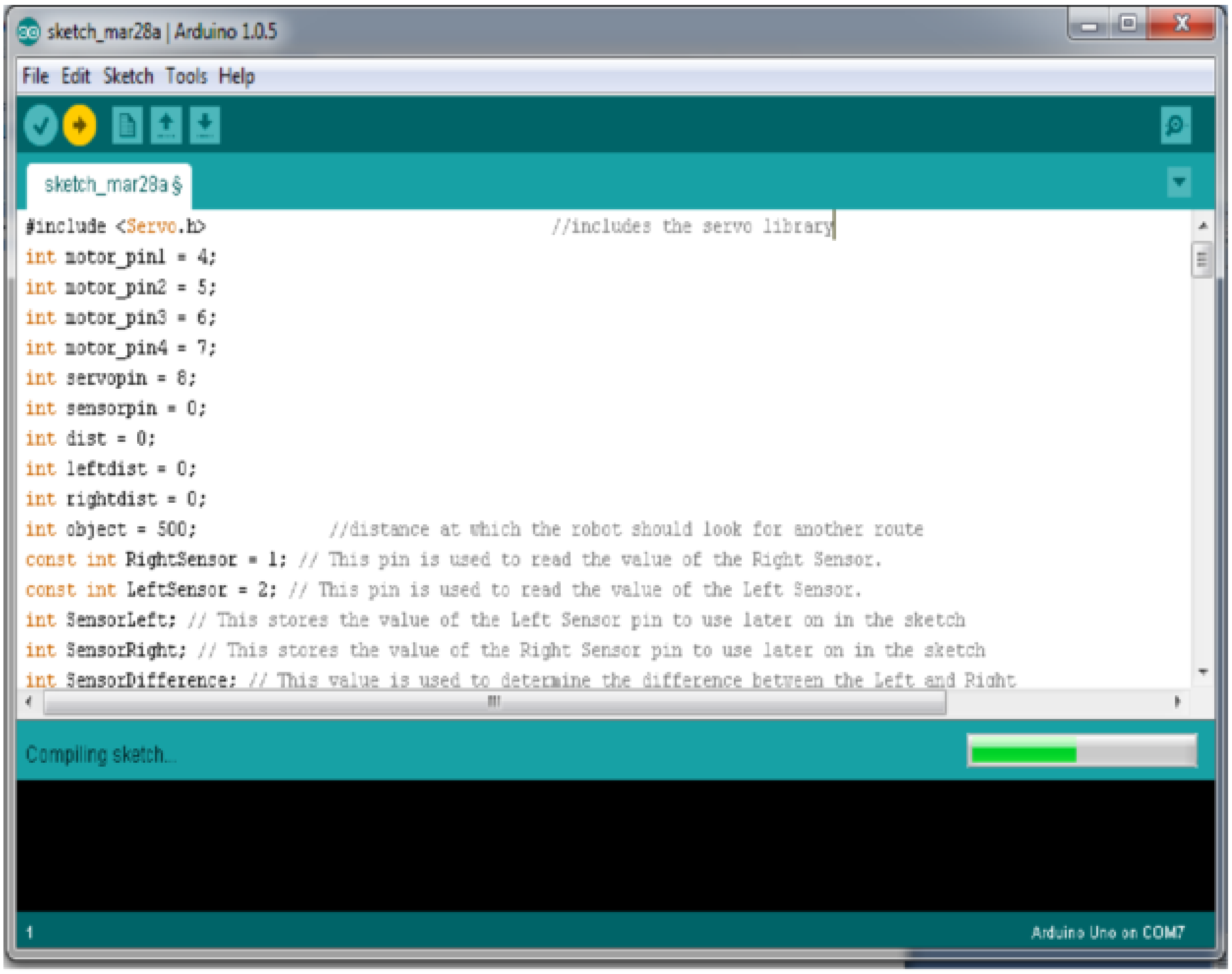

sketch_mar27a | Arduino 1.0.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
sketch_mar27
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
1
Arduino Uno on COM7

4. WRITE THE SKETCH IN ARDUINO IDE



```
#include <Servo.h>                              //includes the servo library
int motor_pin1 = 4;
int motor_pin2 = 5;
int motor_pin3 = 6;
int motor_pin4 = 7;
int servopin = 8;
int sensorpin = 0;
int dist = 0;
int leftdist = 0;
int rightdist = 0;
int object = 500;             //distance at which the robot should look for another route
const int RightSensor = 1; // This pin is used to read the value of the Right Sensor.
const int LeftSensor = 2; // This pin is used to read the value of the Left Sensor.
int SensorLeft; // This stores the value of the Left Sensor pin to use later on in the sketch
int SensorRight; // This stores the value of the Right Sensor pin to use later on in the sketch
int SensorDifference; // This value is used to determine the difference between the Left and Right
```

5. COMPILE AND UPLOAD THE SKETCH TO ARDUINO BOARD



```
#include <Servo.h>                                   //includes the servo library
int motor_pin1 = 4;
int motor_pin2 = 5;
int motor_pin3 = 6;
int motor_pin4 = 7;
int servopin = 8;
int sensorpin = 0;
int dist = 0;
int leftdist = 0;
int rightdist = 0;
int object = 500;             //distance at which the robot should look for another route
const int RightSensor = 1; // This pin is used to read the value of the Right Sensor.
const int LeftSensor = 2; // This pin is used to read the value of the Left Sensor.
int SensorLeft; // This stores the value of the Left Sensor pin to use later on in the sketch
int SensorRight; // This stores the value of the Right Sensor pin to use later on in the sketch
int SensorDifference; // This value is used to determine the difference between the Left and Right
```

CHAPTER-3
GLOBAL SYSTEM FOR MOBILE COMMUNICATION

3.1 Introduction

This is a plug and play GSM Modem with a simple to interface serial interface. Use it to send SMS, make and receive calls, and do other GSM operations by controlling it through simple AT commands from micro controllers and computers. It uses the highly popular SIM900 module for all its operations. It comes with a standard RS232 interface which can be used to easily interface the modem to micro controllers and computers.The modem consists of all the required external circuitry required to start experimenting with the SIM300 module like the power regulation, external antenna, SIM Holder, etc.

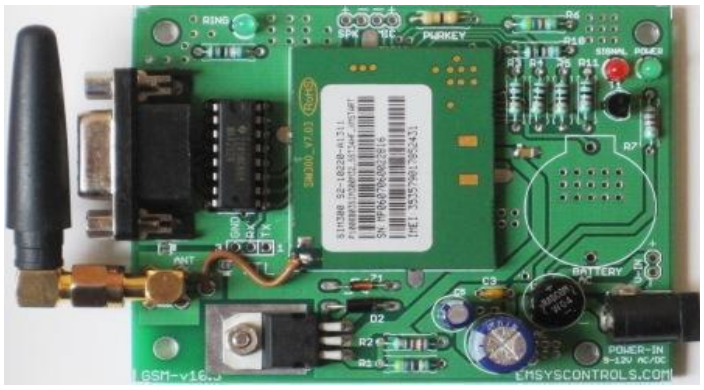


Fig 3.1 GSM (SIM 300) module

3.2 AT Commands:

AT commands are used to control MODEMs. AT is the abbreviation for Attention. These commands come from Hayes commands that were used by the Hayes smart modems. The Hayes commands started with AT to indicate the attention from the MODEM. The dial up and wireless MODEMs (devices that involve machine to machine communication) need AT commands to interact with a computer. These include the Hayes command set as a subset, along with other extended AT commands. AT commands with a GSM/GPRS MODEM or mobile phone can be used to access following information and services:

1. Information and configuration pertaining to mobile device or MODEM and SIM card.
2. SMS services.
3. MMS services.
4. Fax services.
5. Data and Voice link over mobile network.

3.3 EXPLANATION OF COMMONLY USED AT COMMANDS

1) **AT** - This command is used to check communication between the module and the computer.

For example,

```
AT
OK
```

The command returns a result code OK if the computer (serial port) and module are connected properly. If any of module or SIM is not working, it would return a result code ERROR.

2) **+CMGF** - This command is used to set the SMS mode. Either text or PDU mode can be selected by assigning 1 or 0 in the command.

SYNTAX: AT+CMGF=<mode>

0: for PDU mode

1: for text mode

The text mode of SMS is easier to operate but it allows limited features of SMS. The PDU (protocol data unit) allows more access to SMS services but the operator requires bit level knowledge of TPDUs. The headers and body of SMS are accessed in hex format in PDU mode so it allows availing more features.

For example,

```
AT+CMGF=1 10
OK
```

3) **+CMGW** - This command is used to store message in the SIM.

SYNTAX: AT+CMGW=” Phone number”> Message to be stored Ctrl+z

As one types AT+CMGW and phone number, „>‟ sign appears on next line where one can type the message. Multiple line messages can be typed in this case. This is why the message is Terminated by providing a „Ctrl+z‟ combination. As Ctrl+z is pressed, the following information response is displayed on the screen.

+CMGW: Number on which message has been stored

4) **+CMGS -** This command is used to send a SMS message to a phone number.

SYNTAX: AT+CMGS= serial number of message to be send.

As the command AT+CMGS and serial number of message are entered, SMS is sent to the particular SIM.

For example,

AT+CMGS=1

OK

5) **ATD -** This command is used to dial or call a number.

SYNTAX: ATD<Phone number>(Enter)

For example,

ATD123456789

6) **ATA -** This command is used to answer a call. An incoming call is indicated by a message „RING‟ which is repeated for every ring of the call. When the call ends „NO CARRIER‟ is displayed on the screen.

SYNTAX: ATA (Enter)

As ATA followed by enter key is pressed, incoming call is answered.

For example,

RING

RING

ATA

7) **ATH -** This command is used to disconnect remote user link with the GSM module.

SYNTAX: ATH (Enter) 11

3.4 INTERFACING GSM TO 8052

Figure 5.1 shows how to interface the GSM with microcontroller. The GSM module is communicate the microcontroller with mobile phones through UART. To communicate over UART or USART, we just need three basic signals which are namely, RXD (receive), TXD (transmit), GND (common ground).

GSM modem interfacing with microcontroller for SMS control of industrial equipment. The sending SMS through GSM modem when interfaced with microcontroller or PC is much simpler as compared with sending SMS through UART.

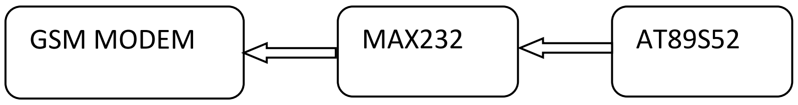


Fig 3.2 Interfacing GSM module with 8052

3.5 AT COMMANDS TO SEND SMS:

There are basically two modes to work with SMS, i.e.:

- PDU (Protocol Data Unit)
- Text mode.

A mobile phone internally uses PDU format. Developers normally uses text mode because it is easier to use. AT+CMGF is the command to set the mode.

AT+CMGF=0

Sets the format to PDU mode.

AT+CMGF=1

sets the format to text mode.

AT+CMGF?

queries the current format.

Sending SMS

The following commands change the message format to text mode and send a text message.

```
AT+CMGF=1
OK
AT+CMGS="9848102222"
> FIRE ACCIDENT <Ctrl>+<Z>
+CMGS: 44
OK
```

CHAPTER 4

GLOBAL POSITIONING SYSTEM

4.1 Introduction

The **Global Positioning System (GPS)** is the only fully functional Global Navigation Satellite System (GNSS). The GPS uses a constellation of between 24 and 32 Medium Earth Orbit satellites that transmit precise microwave signals, which enable GPS receivers to determine their location, speed,. GPS was developed by the United States Department of Defense. Its official name is **NAVSTAR-GPS**. Although NAVSTAR-GPS is not an acronym, a few backronyms have been created for it. The GPS satellite constellation is managed by the United States Air Force 50th Space Wing.

Following the shooting down of Korean Air Lines Flight 007 in 1983, President Ronald Reagan issued a directive making the system available free for civilian use as a common good. Since then, GPS has become a widely used aid to navigation worldwide, and a useful tool for map-making, land surveying, commerce, scientific uses, and hobbies such as geocaching. GPS also provides a precise time reference used in many applications including scientific study of earthquakes, and synchronization of telecommunications networks.

4.2 Basic concept of Gps operation

A GPS receiver calculates its position by carefully timing the signals sent by the constellation of GPS satellites high above the Earth. Each satellite continually transmits messages containing the time the message was sent, a precise orbit for the satellite sending the message (the ephemeris), and the general system health and rough orbits of all GPS satellites (the almanac). These signals travel at the speed of light through outer space, and slightly slower through the atmosphere. The receiver uses the arrival time of each message to measure the distance to each satellite, from which it determines the position of the receiver (conceptually the intersection of spheres - see trilateration) The resulting coordinates are converted to more user-friendly forms such as latitude and longitude, or location on a map, then displayed to the user.

I

4.3 Position calculation introduction

To provide an introductory description of how a GPS receiver works, errors will be ignored in this section. Using messages received from a minimum of four visible satellites, a GPS receiver is able to determine the satellite positions and time sent.

The x, y, and z components of position and the time sent are designated as $[x_i, y_i, z_i, t_i]$where the subscript i denotes the satellite number and has the value 1, 2, 3, or 4. Knowing the indicated time the message was received tr_i, the GPS receiver can compute the indicated transit time, $(tr_i - t_i)$. of the message.

Assuming the message traveled at the speed of light, c, the distance travelled, p_i can be computed as $(tr_i - t_i)\,c$. Knowing the distance from GPS receiver to a satellite and the position of a satellite implies that the GPS receiver is on the surface of a sphere centered at the position of a satellite. Thus we know that the indicated position of the GPS receiver is at or near the intersection of the surfaces of four spheres. In the ideal case of no errors, the GPS receiver will be at an intersection of the surfaces of four spheres. The surfaces of two spheres if they intersect in more than one point intersect in a circle. A figure, two sphere surfaces intersecting in a circle, is shown below.

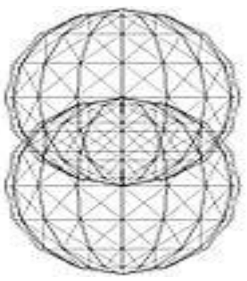


Fig 4.1 Two Sphere Surfaces Intersecting in a Circle

The article, trilateration, shows mathematically that two spheres intersecting in more than one point intersect in a circle.

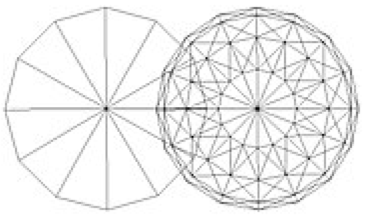


Fig 4.2 Surface of Sphere Intersecting a Circle (not disk) at Two Points

A circle and sphere surface in most cases of practical interest intersects at two points, although it is conceivable that they could intersect in 0 or 1 point.

4.4 Correcting GPS clock

The method of calculating position for the case of no errors has been explained. One of the most important errors is the error in the GPS receiver clock. Because of the very large value of c, the speed of light, the estimated distances from the GPS receiver to the satellites, the pseudo ranges, are very sensitive to errors in the GPS receiver clock.

It is likely the surfaces of the three spheres intersect since the circle of intersection of the first two spheres is normally quite large and thus the third sphere surface is likely to intersect this large circle. It is very unlikely that the surface of the sphere corresponding to the fourth satellite will intersect either of the two points of intersection of the first three since any clock error could cause it to miss intersecting a point. However the distance from the valid estimate of GPS receiver position to the surface of the sphere corresponding to the fourth satellite can be used to compute a clock correction. Let r_4 denote the distance from the valid estimate of GPS receiver position to the fourth satellite and let $p_4$denote the pseudo range of the fourth satellite. Let $da = r_4 - p_4$. Note that $da$is the distance from the computed GPS receiver position to the surface of the sphere corresponding to the fourth satellite. Thus the quotient, $b = da/c$, provides an estimate of:
(correct time) - (time indicated by the receiver's on-board clock) and the GPS receiver clock can be advanced if $b$is positive or delayed if $b$is negative.

4.5 System segmentation

The current GPS consists of three major segments. These are the space segment (SS), a control segment (CS), and a user segment (US).

4.6 Space segment

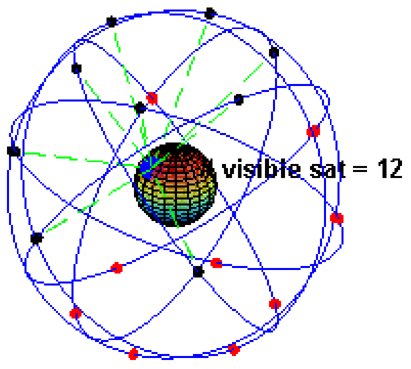
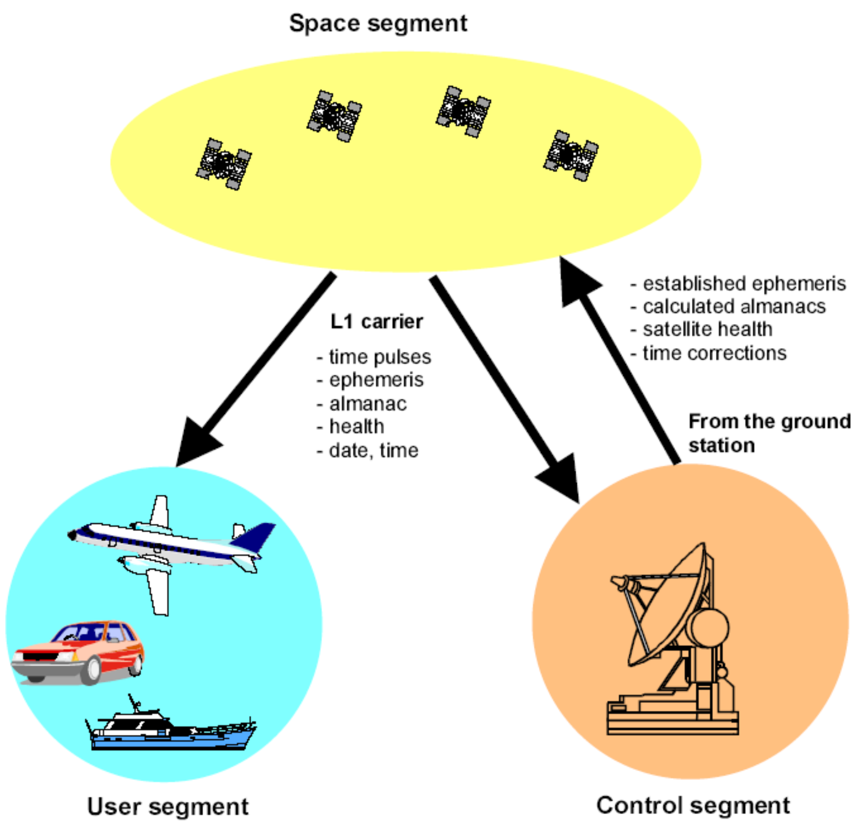


Fig 4.3 A visual example of the GPS constellation in motion with the Earth rotating.



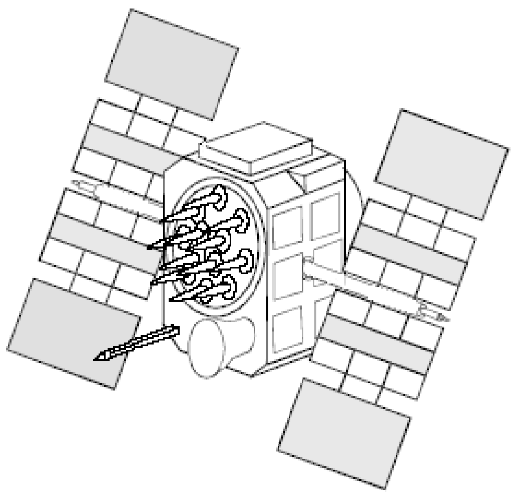


Figure 4.4 : A GPS satellite

4.7 Control segment

The flight paths of the satellites are tracked by US Air Force monitoring stations in Hawaii, Kwajalein, Ascension Island, Diego Garcia, and Colorado Springs, Colorado, along with monitor stations operated by the National Geospatial-Intelligence Agency (NGA). The tracking information is sent to the Air Force Space Command's master control station at Schriever Air Force Base in Colorado Springs, which is operated by the 2nd Space Operations Squadron (2 SOPS) of the United States Air Force (USAF). Then 2 SOPS contacts each GPS satellite regularly with a navigational update (using the ground antennas at Ascension Island, Diego Garcia, Kwajalein, and Colorado Springs). 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.

User segment

PS receivers come in a variety of formats, from devices integrated into cars, phones, and watches, to dedicated devices such as those shown here from manufacturers Trimble, Garmin and Leica (left to right).

The user's GPS receiver is the user segment (US) of the GPS. In general, GPS receivers are composed of an antenna, tuned to the frequencies transmitted by the satellites, receiver-processors, and a highly-stable clock (often a crystal oscillator). They may also include a display for providing location and speed information to the user.

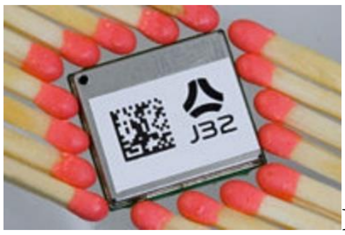


Fig 4.5 A typical OEM GPS receiver module measuring 15×17 mm.

GPS receivers may include an input for differential corrections, using the RTCM SC-104 format. This is typically in the form of a RS-232 port at 4,800 bit/s speed. Data is actually sent at a much lower rate, which limits the accuracy of the signal sent using RTCM. Receivers with internal DGPS receivers can outperform those using external RTCM data.

Fig 4.6 A typical GPS receiver with integrated antenna

.4.8 Navigation signals

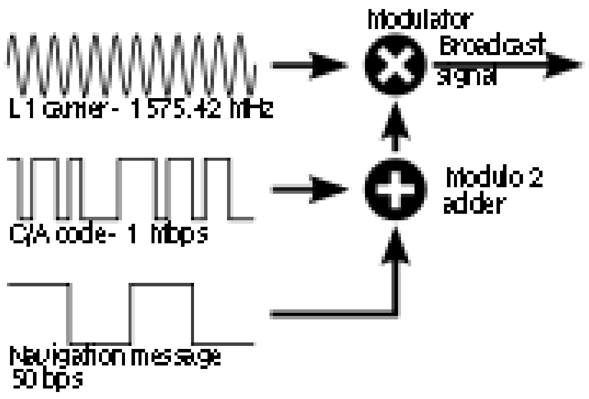


Fig 4.7 GPS broadcast signal

The **almanac** consists of coarse orbit and status information for each satellite in the constellation, an ionosphere model, and information to relate GPS derived time to Coordinated Universal Time (UTC). A new part of the almanac is received for the last 12 seconds in each 30 second frame. Each frame contains 1/25th of the almanac, so 12.5 minutes are required to receive the entire almanac from a single satellite. The almanac serves several purposes. The first is to assist in the acquisition of satellites at power-up by allowing the receiver to generate a list of visible satellites based on stored position and time, while an ephemeris from each satellite is needed to compute position fixes using that satellite. In older hardware, lack of an almanac in a new receiver would cause long delays before providing a valid position, because the search for each satellite was a slow process.

4.9 Position determination

Before providing a more mathematical description of position calculation, the introductory material on these topics is reviewed. To describe the basic concept of how a GPS receiver works, the errors are at first ignored. Using messages received from four satellites, the GPS receiver is able to determine the satellite positions and time sent. The x, y, and z components of position and the time sent are designated as $[x_i, y_i, z_i, t_i]$ where the subscript i denotes which satellite and has the value 1, 2, 3, or 4. Knowing the indicated time the message was received tr_i, the GPS receiver can compute the indicated transit time, $(tr_i - t_i)$. of the message. Assuming the message traveled at the speed of light, c, the distance traveled, $p_i$can be computed as $(tr_i - t_i)\,c$. Knowing the distance from GPS receiver to a satellite and the position of a satellite implies that the GPS receiver is on the surface of a sphere centered at the position of a satellite. Thus we know that the indicated position of the GPS receiver is at or near the intersection of the surfaces of four spheres. In the ideal case of no errors, the GPS receiver will be at an intersection of the surfaces of four spheres. The surfaces of two spheres if they intersect in more than one point intersect in a circle. A figure, Two Sphere Surfaces Intersecting in a Circle, is shown below depicting this which hopefully will aid the reader in visualizing this intersection.

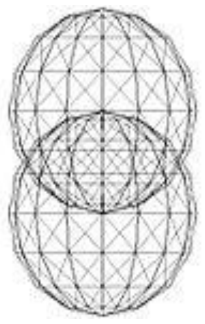


Fig 4.8 Two Sphere Surfaces Intersecting in a Circle

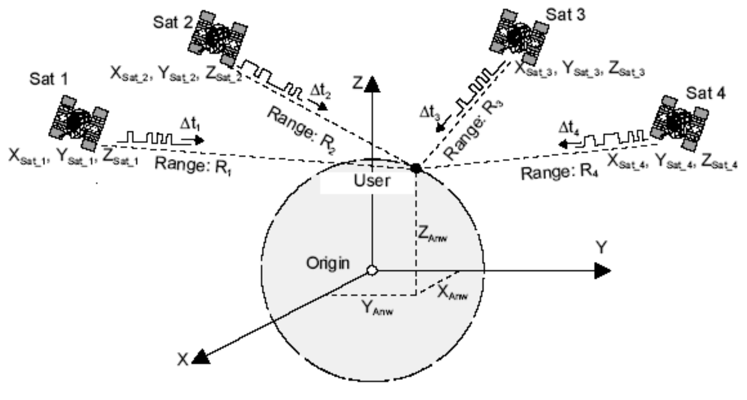

Sat 2
Sat 3
Sat 1
X_{Sat_2}, Y_{Sat_2}, Z_{Sat_2}
X_{Sat_3}, Y_{Sat_3}, Z_{Sat_3}
Sat 4
Δt_2
Z
Δt_3
Range: R_2
Range: R_3
Δt_1
Δt_4
X_{Sat_1}, Y_{Sat_1}, Z_{Sat_1}
Range: R_1
Range: R_4
X_{Sat_4}, Y_{Sat_4}, Z_{Sat_4}
User
Z_{Anw}
Y
Origin
X_{Anw}
Y_{Anw}
X

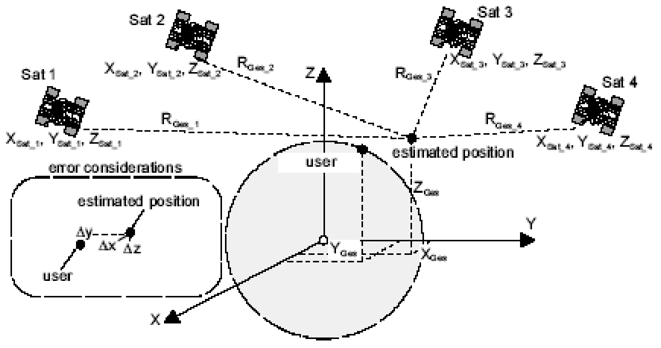
Fig 4.9 Three dimensional co-ordinate system

Generally (with $\Delta x = x - x_0$): $f(x) = f(x_0) + \frac{f'}{1!}(x_0) \cdot \Delta x + \frac{f''}{2!}(x_0)^2 \cdot \Delta x + \frac{f'''}{3!}(x_0)^3 \cdot \Delta x + \ldots$

Simplified (1st part only): $f(x) = f(x_0) + f'(x_0) \cdot \Delta x$ (7a)

In order to linearise the four equations (6a), an arbitrarily estimated value x_0 must therefore be incorporated in the vicinity of x.

For the GPS system, this means that instead of calculating X_{Anw}, Y_{Anw} and Z_{Anw} directly, an estimated position X_{Ges}, Y_{Ges} and Z_{Ges} is initially used (Figure 23).



The article, trilateration, shows mathematically how the equation for a circle is determined. A circle and sphere surface in most cases of practical interest intersects at two points, although it is conceivable that they could intersect in 0 or 1 point. Another figure, Surface of Sphere Intersecting a Circle (not disk) at Two Points, is shown below to aid in visualizing this intersection. Again trilateration clearly show this mathematically. The correct position of the GPS receiver is the one that is closest to the fourth sphere.

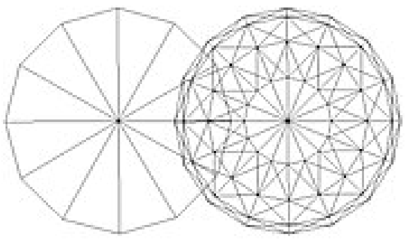


Fig 4.10 Surface of Sphere Intersecting a Circle (not disk) at Two Points

Let $b$denote the clock error or bias, the amount by which the receiver's clock is slow. The GPS receiver has four unknowns, the three components of GPS receiver position and the clock bias $[x, y, z, b]$. The equation of the sphere surfaces are given by:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = \left((tr_i + b - t_i)c\right)^2, \ i = 1, 2, 3, 4.$$

Another useful form of these equations is in terms of the *pseudoranges*, which are simply the ranges approximated based on GPS receiver clock's indicated (i.e. uncorrected) time so that $p_i = (tr_i - t_i)\,c$.

Then the equations becomes:

$$p_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2)} - bc,\ \ i = 1, 2, 3, 4.$$

Two of the most important methods of computing GPS receiver position and clock bias are (1) trilateration followed by one dimensional numerical root finding and (2) multidimensional Newton-Raphson. These two methods along with their advantages are discussed.

- Solve by trilateration followed by one dimensional numerical root finding. This method involves using Trilateration to determine the intersection of the surfaces of three spheres. It is clearly shown in trilateration that the surfaces of three spheres intersect in 0, 1, or 2 points. In the usual case of two intersections, the solution which is nearest the surface of the sphere corresponding to the fourth satellite is chosen. The surface of the earth can also sometimes be used instead, especially in the case of civilian GPS receivers since it is illegal in the United States to track vehicles of more than 60,000 feet in altitude. The bias, $b$is then computed based on the distance from the solution to the surface of the sphere corresponding to the fourth satellite. Using an updated received time based on this bias, new spheres are computed and the process is repeated. One advantage of this method is that it involves one dimensional as opposed to multidimensional numerical root finding.

- Utilize multidimensional Newton-Raphson. Linearize around an approximate solution say $\left[x^{(k)}, y^{(k)}, z^{(k)}, b^{(k)}\right]$from iteration k, then solve four linear equations derived from the quadratic equations above to obtain $\left[x^{(k+1)}, y^{(k+1)}, z^{(k+1)}, b^{(k+1)}\right]$.
- The radii are large and so the sphere surfaces are close to flat. This near flatness may cause the iterative procedure to converge rapidly in the case where $b$is near the correct value and the primary change is in the values of $x, y,\ and\ z$, since in this case the problem is merely to find the intersection of nearly flat surfaces and thus close to a linear problem. However when $b$is changing significantly, this near flatness does not appear to be advantageous in producing rapid convergence, since in this case these near flat surfaces will be moving as the spheres expand and contract.
- This possible fast convergence is an advantage of this method. Also it has been claimed that this method is the "typical" method used by GPS receivers. A disadvantage of this method is that according to, "There are no good general methods for solving systems of more than one nonlinear equation."

4.10 P(Y) code

Calculating a position with the P(Y) signal is generally similar in concept, assuming one can decrypt it. The encryption is essentially a safety mechanism: if a signal can be successfully decrypted, it is reasonable to assume it is a real signal being sent by a GPS satellite. In comparison, civil receivers are highly vulnerable to spoofing since correctly formatted C/A signals can be generated using readily available signal generators. RAIM features do not protect against spoofing, since RAIM only checks the signals from a navigational perspective.

4.11 Applications

The Global Positioning System, while originally a military project is considered a *dual-use* technology, meaning it has significant applications for both the military and the civilian industry.

4.11 a) Military

The military applications of GPS span many purposes:

- Navigation: GPS allows soldiers to find objectives in the dark or in unfamiliar territory, and to coordinate the movement of troops and supplies. The GPS-receivers commanders and soldiers use are respectively called the **Commanders Digital Assistant** and the **Soldier Digital Assistant**.
- Target tracking: Various military weapons systems use GPS to track potential ground and air targets before they are flagged as hostile. These weapon systems pass GPS co-ordinates of targets to precision-guided munitions to allow them to engage the targets accurately.
- Missile and projectile guidance: GPS allows accurate targeting of various military weapons including **ICBMs**, cruise missiles and precision-guided munitions. Artillery projectiles with embedded GPS receivers able to withstand accelerations of 12,000G have been developed for use in 155 mm howitzers.
- Search and Rescue: Downed pilots can be located faster if they have a GPS receiver.
- Reconnaissance and Map Creation: The military use GPS extensively to aid mapping and reconnaissance.
- The GPS satellites also carry a set of nuclear detonation detectors consisting of an optical sensor (Y-sensor), an X-ray sensor, a dosimeter, and an Electro-Magnetic Pulse (EMP) sensor (W-sensor) which form a major portion of the United States Nuclear Detonation Detection System.

4.11 b)Civilian

This antenna is mounted on the roof of a hut containing a scientific experiment needing precise timing. Many civilian applications benefit from GPS signals, using one or more of three basic components of the GPS: absolute location, relative movement, and time transfer.

The ability to determine the receiver's absolute location allows GPS receivers to perform as a surveying tool or as an aid to navigation. The capacity to determine relative movement enables a receiver to calculate local velocity and orientation, useful in vessels or observations of the Earth. Being able to synchronize clocks to exacting standards enables time transfer, which is critical in large communication and observation systems. An example is CDMA digital cellular. Each base station has a GPS timing receiver to synchronize its spreading codes with other base stations to facilitate inter-cell hand off and support hybrid GPS/CDMA positioning of mobiles for emergency calls and other applications. Finally, GPS enables researchers to explore the Earth environment including the atmosphere, ionosphere and gravity field. GPS survey equipment has revolutionized tectonics by directly measuring the motion of faults in earthquakes.

4.12 GPS Module

Latitude and longitude are usually provided in the geodetic datum on which GPS is based (WGS-84).

- ❖ Receivers can often be set to convert to other user-required datums.
- ❖ Receiver position is computed from the SV positions, the measured pseudo-ranges, and a receiver position estimate.
- ❖ Four satellites allow computation of three position dimensions and time.

- ❖ Three satellites could be used determine three position dimensions with a perfect receiver clock.
- ❖ In practice this is rarely possible and three SVs are used to compute a two-dimensional, horizontal fix (in latitude and longitude) given an assumed height.
- ❖ This is often possible at sea or in altimeter equipped aircraft.
- ❖ Five or more satellites can provide position, time and redundancy.
- ❖ Twelve channel receivers allow continuous tracking of all available satellites, including tracking of satellites with weak or occasionally obstructed signals.

4.13 GPS Receivers

GPS receivers require different signals in order to function figure. These variables are broadcast after position and time have been successfully calculated and determined. To ensure that the different types of appliances are portable there are either international standards for data exchange (NMEA and RTCM), or the Manufacturer provides defined (proprietary) formats and protocols.

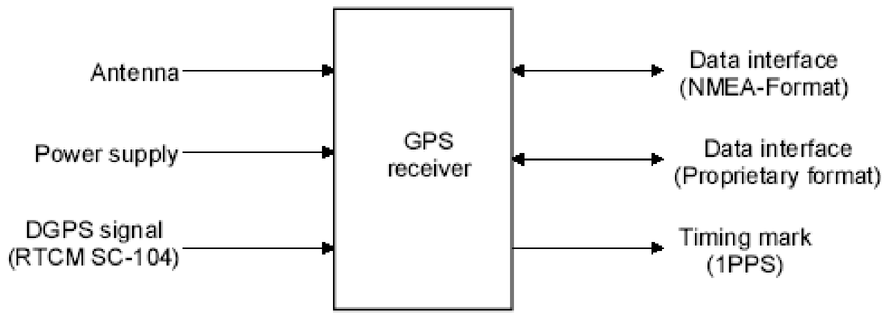

Antenna
Power supply
DGPS signal (RTCM SC-104)
GPS receiver
Data interface (NMEA-Format)
Data interface (Proprietary format)
Timing mark (1PPS)

Fig 4.11 Block diagram of a GPS receiver with interfaces

The following seven datasets are widely used with GPS module store lay GPS information [xv]:

1. GGA (GPS Fix Data, fixed data for the Global Positioning System)

2. GLL (Geographic Position–Latitude/Longitude)

3. GSA (GNSSDOP and Active Satellites, degradation of accuracy and the number of active satellites in the Global Satellite Navigation System)

4. GSV (GNSSSatellitesinView, satellites in view in the Global Satellite Navigation System)

5. RMC (Recommended Minimum Specific GNSSD at a)

6. VTG (Course over Ground and Ground Speed, horizontal course and horizontal velocity)

7. ZDA (Time & Date) structure of the NMEA protocol in the case of NMEA, there at at which data is transmitted is 4800 Baud using printable 8bit ASCII characters. transmission begin switch start bit (logical zero), followed by eight data bits and as to bit (logical one)added at the end. No parity bits are used.

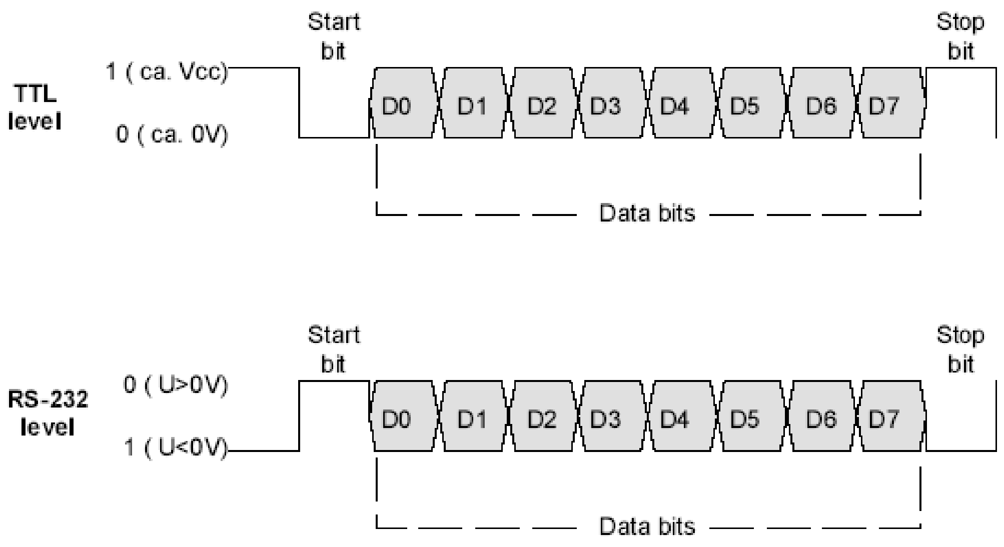


Fig 4.11 NMEA format (TTL and RS-232 level)

The different levels must be take into consideration depending on whether the GPS receiver used has a TTL or RS-232interface (Figure)

• In the case of a TTL level interface, a logical zero corresponds to approx. 0V and a logical one roughly to the operating voltage of the system (+3.3V.+5V)

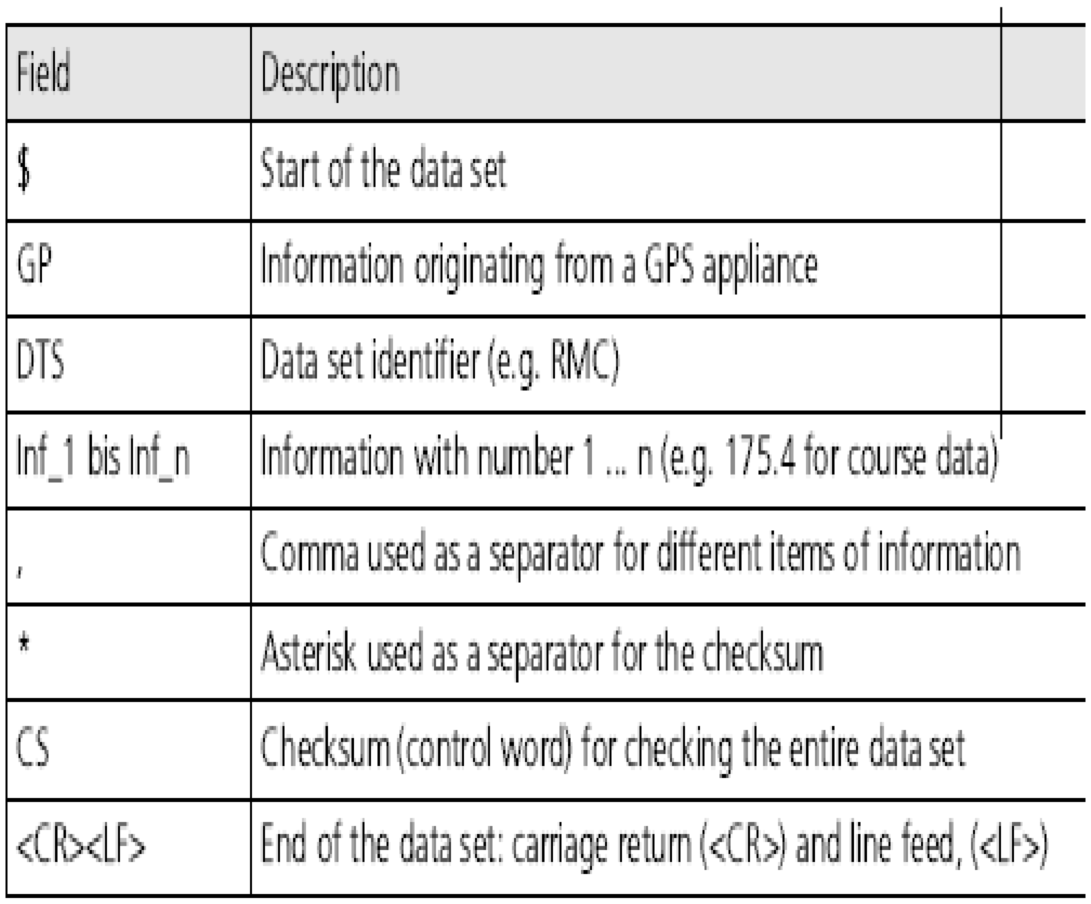
• In the case of an RS232 interface a logical zero corresponds to a positive voltage (+3V_._+15V)and a Logical one a negative voltage(-3V...–15V).

If a GPS module with a TTL level interface is connected to an appliance with an RS-232 interface, a level conversion must b effected A few GPS modules all the baud rate to be increased (upto38400bitsper_second).

Each GPS data set is formed in the same way and has the following structure:

$GPDTS, Inf1,Inf2,Inf3,Inf4,Inf5,Inf6,Infn*CS<CR><LF>

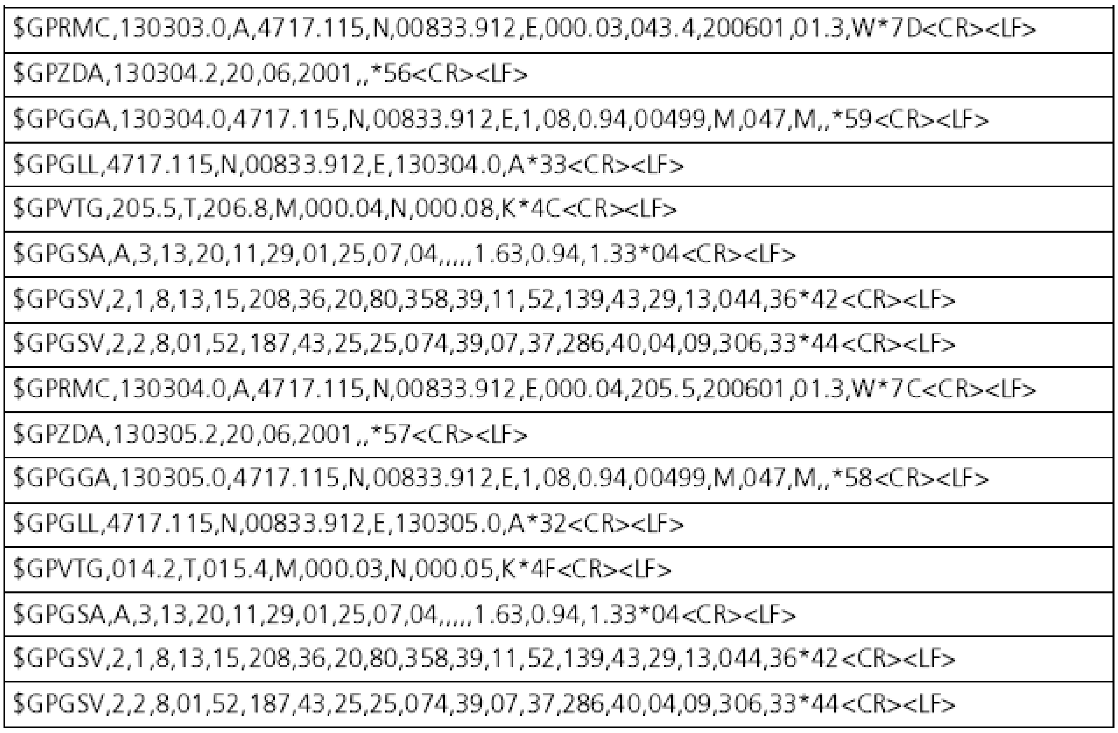
The function of the individual characters or character set is explained inTable8



Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
DTS	Data set identifier (e.g. RMC)
Inf_1 bis Inf_n	Information with number 1 ... n (e.g. 175.4 for course data)
,	Comma used as a separator for different items of information
*	Asterisk used as a separator for the checksum
CS	Checksum (control word) for checking the entire data set
<CR><LF>	End of the data set: carriage return (<CR>) and line feed, (<LF>)

Table 1: Description of the individual NMEA DATA SET block

The following NMEA Protocol was recorded using a GPS receiver (Table2):



$GPRMC,130303.0,A,4717.115,N,00833.912,E,000.03,043.4,200601,01.3,W*7D<CR><LF>
$GPZDA,130304.2,20,06,2001,,*56<CR><LF>
$GPGGA,130304.0,4717.115,N,00833.912,E,1,08,0.94,00499,M,047,M,,*59<CR><LF>
$GPGLL,4717.115,N,00833.912,E,130304.0,A*33<CR><LF>
$GPVTG,205.5,T,206.8,M,000.04,N,000.08,K*4C<CR><LF>
$GPGSA,A,3,13,20,11,29,01,25,07,04,,,,,1.63,0.94,1.33*04<CR><LF>
$GPGSV,2,1,8,13,15,208,36,20,80,358,39,11,52,139,43,29,13,044,36*42<CR><LF>
$GPGSV,2,2,8,01,52,187,43,25,25,074,39,07,37,286,40,04,09,306,33*44<CR><LF>
$GPRMC,130304.0,A,4717.115,N,00833.912,E,000.04,205.5,200601,01.3,W*7C<CR><LF>
$GPZDA,130305.2,20,06,2001,,*57<CR><LF>
$GPGGA,130305.0,4717.115,N,00833.912,E,1,08,0.94,00499,M,047,M,,*58<CR><LF>
$GPGLL,4717.115,N,00833.912,E,130305.0,A*32<CR><LF>
$GPVTG,014.2,T,015.4,M,000.03,N,000.05,K*4F<CR><LF>
$GPGSA,A,3,13,20,11,29,01,25,07,04,,,,,1.63,0.94,1.33*04<CR><LF>
$GPGSV,2,1,8,13,15,208,36,20,80,358,39,11,52,139,43,29,13,044,36*42<CR><LF>
$GPGSV,2,2,8,01,52,187,43,25,25,074,39,07,37,286,40,04,09,306,33*44<CR><LF>

Recording of an NMEA protocol

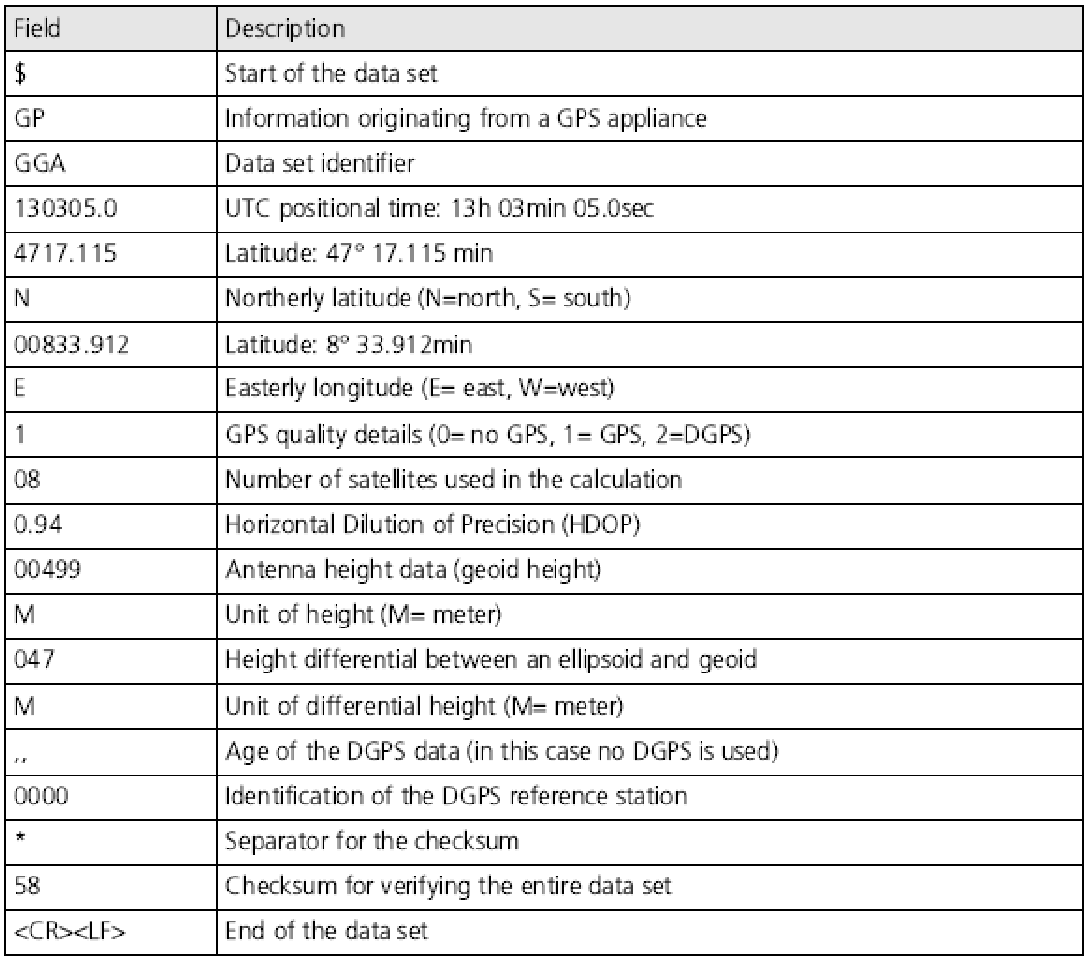
4.14 GGA data set

The GGA dataset (GPS Fix Data) contains information on time, longitude and latitude, the quality of the system, The number of satellites used and the height.

An example of a GGA dataset:

$GPGGA,
130305.0,4717.115,N,00833.912,E,1,08,0.94,00499,M,047,M,,*58<CR><LF>

The function of the individual characters or character sets is explained in Table3



Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
GGA	Data set identifier
130305.0	UTC positional time: 13h 03min 05.0sec
4717.115	Latitude: 47° 17.115 min
N	Northerly latitude (N=north, S= south)
00833.912	Latitude: 8° 33.912min
E	Easterly longitude (E= east, W=west)
1	GPS quality details (0= no GPS, 1= GPS, 2=DGPS)
08	Number of satellites used in the calculation
0.94	Horizontal Dilution of Precision (HDOP)
00499	Antenna height data (geoid height)
M	Unit of height (M= meter)
047	Height differential between an ellipsoid and geoid
M	Unit of differential height (M= meter)
,,	Age of the DGPS data (in this case no DGPS is used)
0000	Identification of the DGPS reference station
*	Separator for the checksum
58	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual GGA data set blocks

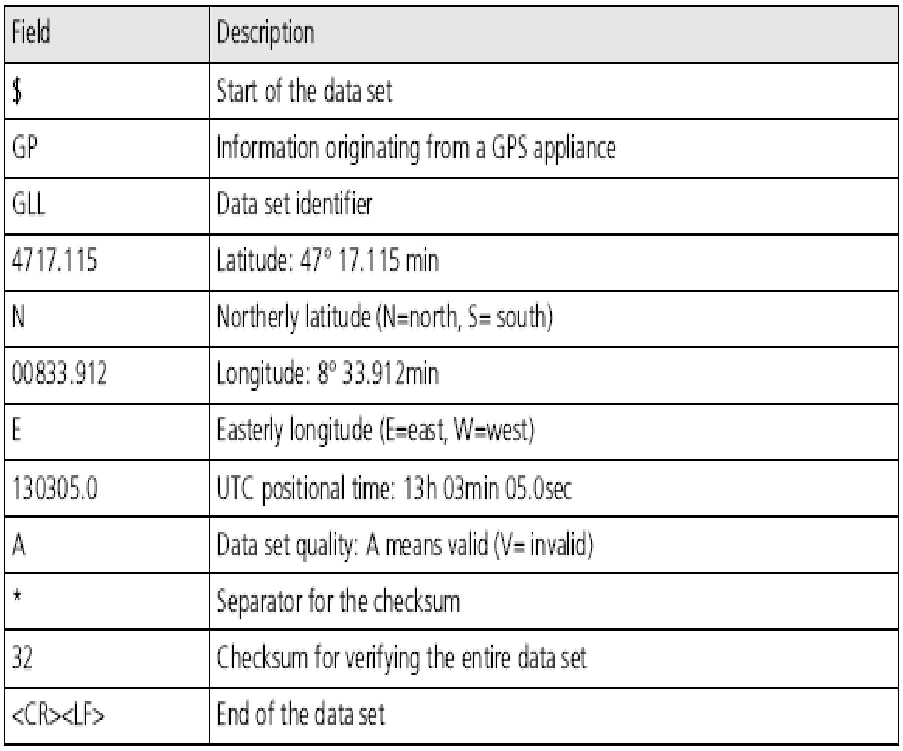
4.15 GLL data set

The GLL dataset (geographic position–latitude/longitude) contains information on latitude and longitude, time, and health.

Example of a GLL data set:

$GPGLL,4717.115,N,00833.912,E,130305.0,A*32<CR><LF>

The function of the individual characters or character sets is explained in Table 4



Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
GLL	Data set identifier
4717.115	Latitude: 47° 17.115 min
N	Northerly latitude (N=north, S= south)
00833.912	Longitude: 8° 33.912min
E	Easterly longitude (E=east, W=west)
130305.0	UTC positional time: 13h 03min 05.0sec
A	Data set quality: A means valid (V= invalid)
*	Separator for the checksum
32	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual GGL data set blocks

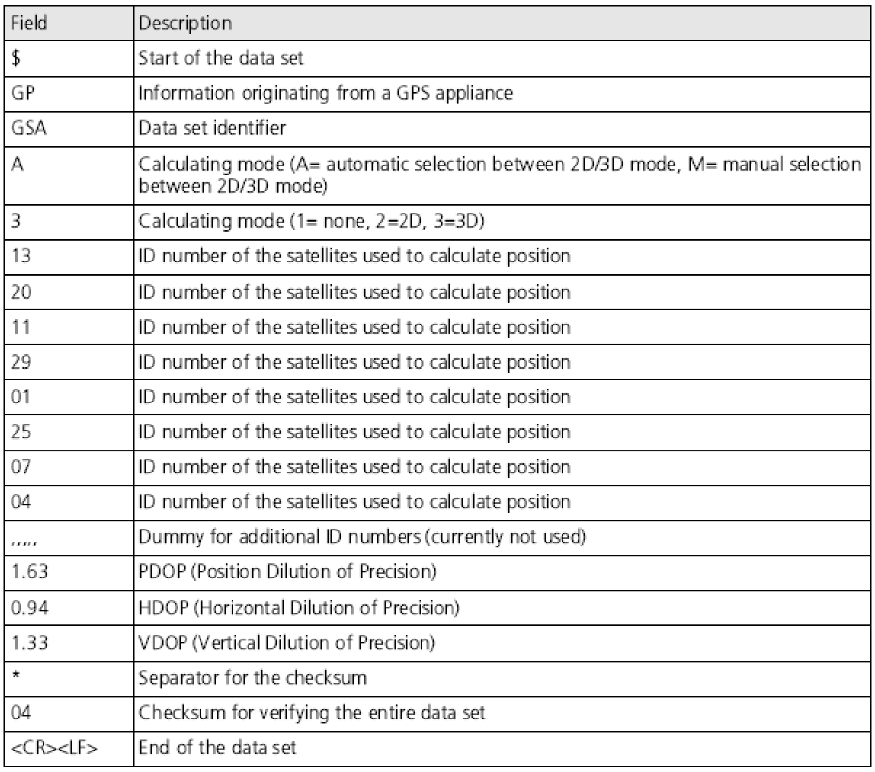
4.16 GSA data set

The GSA dataset (GNSS DOP and Active Satellites) contains information on the measuring mode(2Dor3D),the number of satellites used to determine the position and the accuracy of the measurements (DOP: Dilution of Precision).

An example of a GS A dataset:

$GPGSA,A,3,13,20,11,29,01,25,07,04,,,,1.63,0.94,1.33*04<CR><LF>

The function of the individual character or sets of characters is described in Table 5



Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
GSA	Data set identifier
A	Calculating mode (A= automatic selection between 2D/3D mode, M= manual selection between 2D/3D mode)
3	Calculating mode (1= none, 2=2D, 3=3D)
13	ID number of the satellites used to calculate position
20	ID number of the satellites used to calculate position
11	ID number of the satellites used to calculate position
29	ID number of the satellites used to calculate position
01	ID number of the satellites used to calculate position
25	ID number of the satellites used to calculate position
07	ID number of the satellites used to calculate position
04	ID number of the satellites used to calculate position
,,,,,	Dummy for additional ID numbers (currently not used)
1.63	PDOP (Position Dilution of Precision)
0.94	HDOP (Horizontal Dilution of Precision)
1.33	VDOP (Vertical Dilution of Precision)
*	Separator for the checksum
04	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual GSA data set blocks

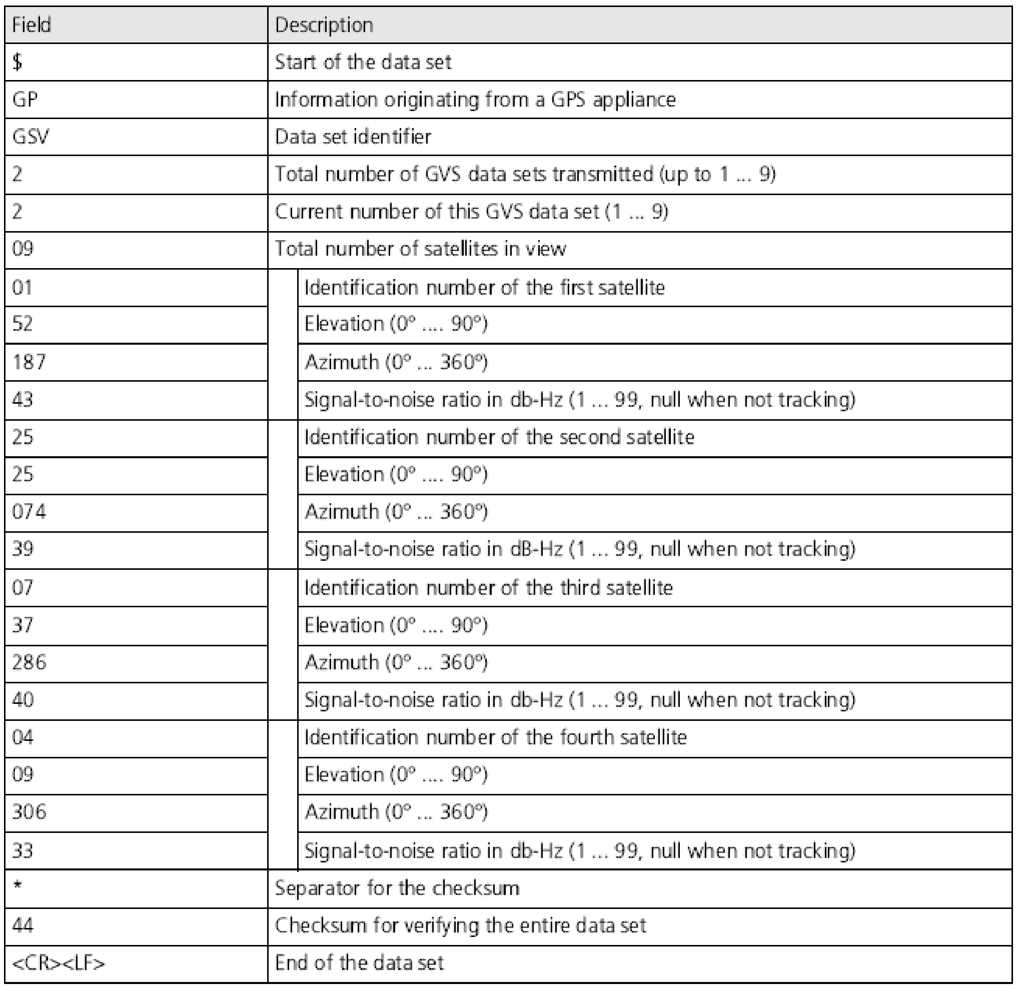
4.17 GSV data set

The GSV data set (GNSS Satellites in View) contains information on the number of satellites in view, there Identification, their elevation and azimuth, and the signal-to-noise ratio.

An example of a GSV dataset:_

$GPGSV,2,2,8,01,52,187,43,25,25,074,39,07,37,286,40,04,09,306,33*44<CR><LF>

The function of the individual characters or character sets is explained in Table 6.



Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
GSV	Data set identifier
2	Total number of GVS data sets transmitted (up to 1 ... 9)
2	Current number of this GVS data set (1 ... 9)
09	Total number of satellites in view
01	Identification number of the first satellite
52	Elevation (0° .... 90°)
187	Azimuth (0° ... 360°)
43	Signal-to-noise ratio in db-Hz (1 ... 99, null when not tracking)
25	Identification number of the second satellite
25	Elevation (0° .... 90°)
074	Azimuth (0° ... 360°)
39	Signal-to-noise ratio in dB-Hz (1 ... 99, null when not tracking)
07	Identification number of the third satellite
37	Elevation (0° .... 90°)
286	Azimuth (0° ... 360°)
40	Signal-to-noise ratio in db-Hz (1 ... 99, null when not tracking)
04	Identification number of the fourth satellite
09	Elevation (0° .... 90°)
306	Azimuth (0° ... 360°)
33	Signal-to-noise ratio in db-Hz (1 ... 99, null when not tracking)
*	Separator for the checksum
44	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual GSV data set blocks

4.18 RMC data set

The RMC data set (Recommended Minimum Specific GNSS) contains information on time, latitude, longitude And height, system status, speed, course and date. This data set is relayed by all GPS receivers.

An example of an RMC data set :_

$GPRMC,130304.0,A,4717.115,N,00833.912,E,000.04,205.5,200601,01.3,W*7C<CR><LF>

The function of the individual characters or character sets is explained in Table 7

Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
RMC	Data set identifier
130304.0	Time of reception (world time UTC): 13h 03 min 04.0 sec
A	Data set quality: A signifies valid (V= invalid)
4717.115	Latitude: 47° 17.115 min
N	Northerly latitude (N=north, S= south)
00833.912	Longitude: 8° 33.912 min
E	Easterly longitude (E=east, W=west)
000.04	Speed: 0.04 knots
205.5	Course: 205.5°
200601	Date: 20th June 2001
01.3	Adjusted declination: 1.3°
W	Westerly direction of declination (E = east)
*	Separator for the checksum
7C	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual RMC data set blocks

The VGT dataset (Course over Ground and Ground Speed) contains information on course and speed.

AnexampleofaVTGdataset:

$GPVTG,014.2,T,015.4,M,000.03,N,000.05,K*4F<CR><LF>

The function of the individual characters or character sets is explained in Table 8

Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
VTG	Data set identifier
014.2	Course 14.2° (T) with regard to the horizontal plane
T	Angular course data relative to the map
015.4	Course 15.4° (M) with regard to the horizontal plane
M	Angular course data relative to magnetic north
000.03	Horizontal speed (N)
N	Speed in knots
000.05	Horizontal speed (Km/h)
K	Speed in km/h
*	Separator for the checksum
4F	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual VTG data set blocks

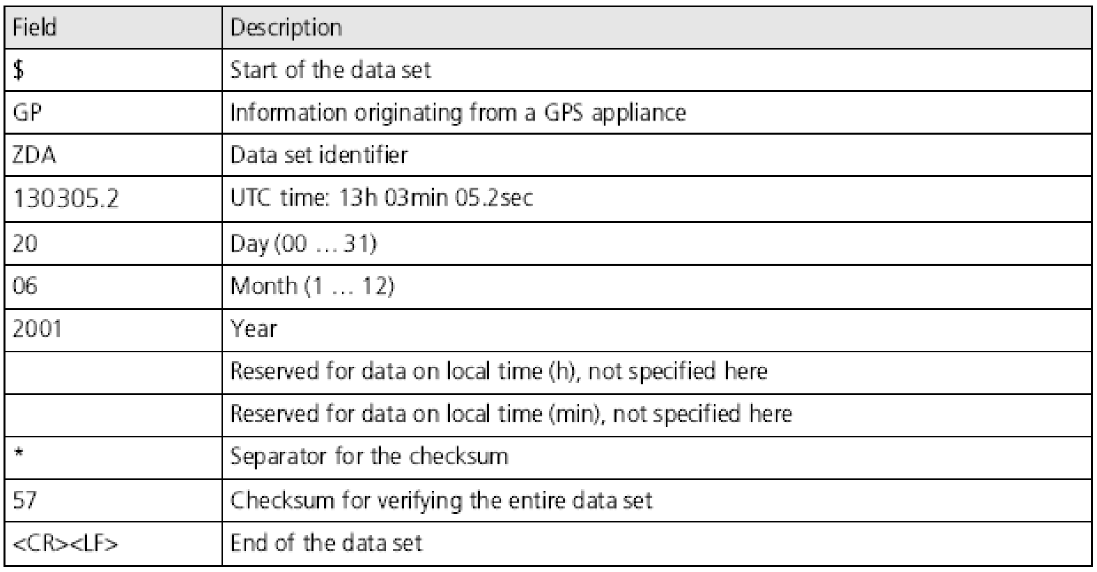
4.19 ZDA data set

The ZDA dataset (time and date)contains information on UTC time, the date and local time.

AnexampleofaZDAdataset:

$GPZDA,130305.2,20,06,2001,,*57<CR><LF>

The function of the individual characters or character sets is explained in Table 9



Field	Description
$	Start of the data set
GP	Information originating from a GPS appliance
ZDA	Data set identifier
130305.2	UTC time: 13h 03min 05.2sec
20	Day (00 … 31)
06	Month (1 … 12)
2001	Year
	Reserved for data on local time (h), not specified here
	Reserved for data on local time (min), not specified here
*	Separator for the checksum
57	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Description of the individual ZDA data set blocks

Calculating the checksum:

The checksum is determined by an exclusive or operation in volving all 8 data bits (excluding start and stop bits)From all transmitted characters, including separators. The exclusive or operation commences after the start of the data set($sign) and ends before the check sum separator(asterisk:*).The8-bit result is divided into 2 sets of 4 bits (nibbles) and each nibble is converted into the appropriate Hexadecimal value(0...9,A...F).The check sum consists of the two hexadecimal values converted in to ASCII characters.

The principle of check sum calculation can be explained with the help of a brief example:

The following NMEA data set has been received and the checksum (CS)must be verified for its correctness.

$GPRTE,1,1,c,0*07 (07 is_the_check sum)

Procedure:

1. Only the characters between $and*are included in the analysis : GPRTE,1,1,c,0

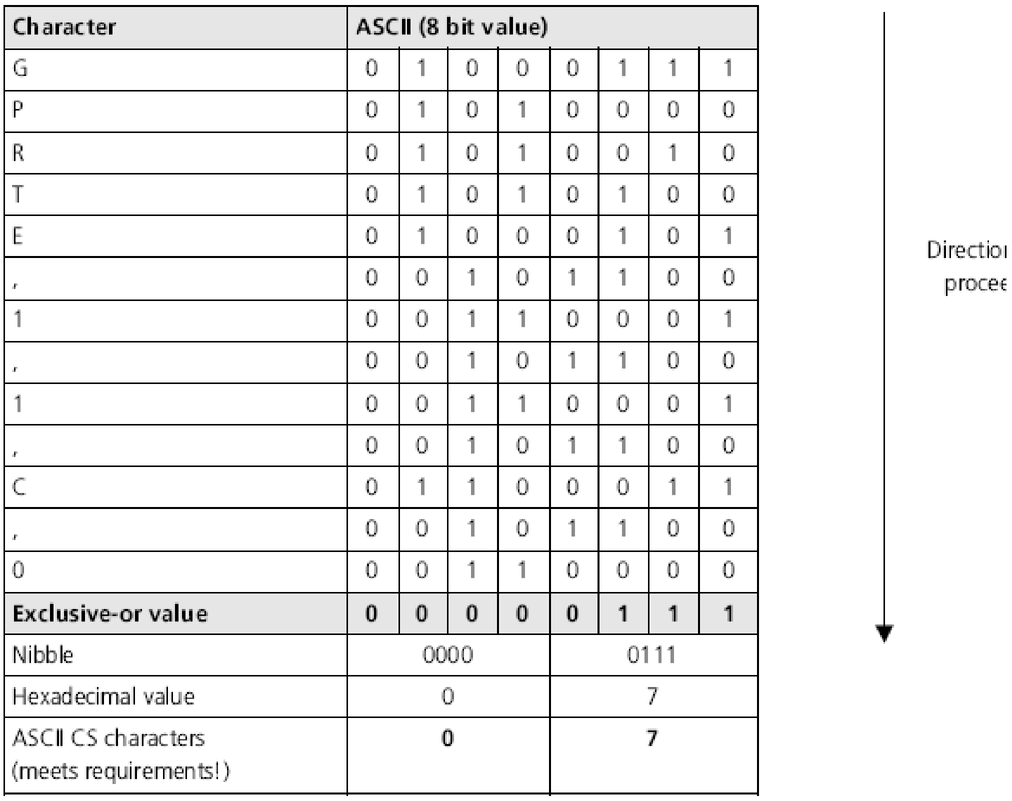
2.These 13 ASCII characters are converted into 8 bit values(see Table)

3.Each individual bit of the 13 ASCII characters is linked to an exclusive-or operation (N.B. If the number of ones is uneven, the exclusive-or value is one)

4. The result is divided in to two nibbles

5. The hexadecimal value of each nibble is determined

6.BothhexadecimalcharactersaretransmittedasASCIIcharacterstoformthechecksum


Directio
procee

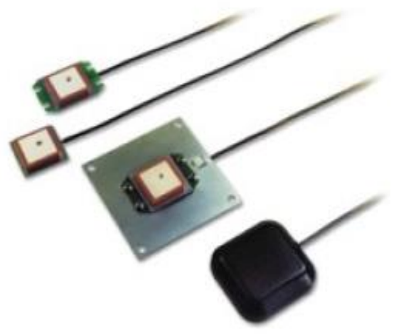
Character	ASCII (8 bit value)							
G	0	1	0	0	0	1	1	1
P	0	1	0	1	0	0	0	0
R	0	1	0	1	0	0	1	0
T	0	1	0	1	0	1	0	0
E	0	1	0	0	0	1	0	1
,	0	0	1	0	1	1	0	0
1	0	0	1	1	0	0	0	1
,	0	0	1	0	1	1	0	0
1	0	0	1	1	0	0	0	1
,	0	0	1	0	1	1	0	0
C	0	1	1	0	0	0	1	1
,	0	0	1	0	1	1	0	0
0	0	0	1	1	0	0	0	0
Exclusive-or value	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
Nibble	0000				0111			
Hexadecimal value	0				7			
ASCII CS characters (meets requirements!)	**0**				**7**			

Table 10 Determining the checksum in the case of NMEA data sets

4.20 Hardware interfaces

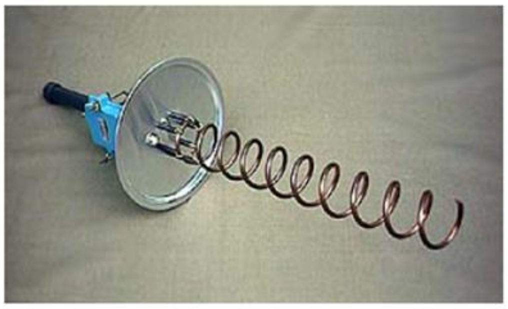
i) Antenna

GPS module scan either be operated with a passive or active antenna. Active antenna, i.e. with a built in pr e amplifier (LNA: Low Noise Amplifier) are powered from the GPS module, the current being provided by the HF signal line. For mobile navigation a l purposes combined antennae (e.g. GSM/FM and GPS) are supplied. GPS antennae receive right-handed circular polarized waves.Two types of antenna are obtain able on the market, Patch antennae and Helix antennae. Patch antenna e are flat, generally have a ceramic and metallised body and are mounted on a metal base plate. In order to ensure a sufficiently high degree of selectivity, the base to Patch surface ratio has to be adjusted. Patch antennae are often cast in a housing (Figure), [xxii]).



Open and cast Patch antennae

Basic structural shape of a Helix antennae



4.21 power Supply

GPS modules must be powered from an external voltage source of 3.3V to 6Volts. In each case, the power draw is very different.

Time pulse: 1PPS and time systems

Most GPS modules generate a time pulse every second, referred to as 1 PPs (1 pulse per second), which is Synchronized to UTC. This signal usually has a TTL level (Figure).

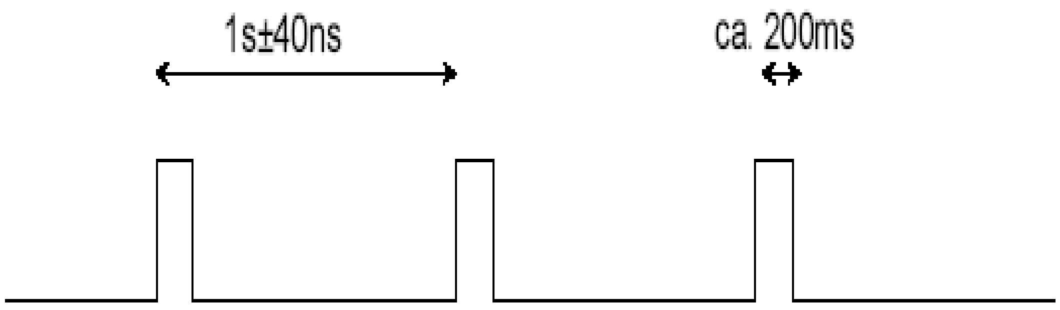


Fig 4.12 1PPS signal

The time pulse can be used to synchronise communication networks (Precision Timing). As time can play a fundamental part when GPS is used to determine a position, a distinction is drawn here

Between five important GPS time systems :

4.21.1 Atomic time (TAI)

The International AtomicTime Scale (Temps Atomique International) was introduced in order to provide a universal 'absolute' time scale that would meet various practical demands and at the same time also be of significanceforGPSpositioning.Since1967, the second has been defined by an atomic constant in physics, the non-radioactive element Cesium 133Cs being selected as a reference. The resonant frequency between the Selected energy states of this atom has been determined at 9 192 631 770 Hz. Time defined in this way is Therefore part of the SI system (System International).The start of atomic time took place on 01.01.1958 at 00.00hours.

4.21.2 Universal time co-ordinate (UTC)

UTC (Universal Time Coordinated)was introduced, in order to have a practical time scale that was orientedTowards universal atomic time and, at the same time, adjusted to universal co -ordinated time. It is distinguished from TAI in the way the seconds are counted, i.e. UTC= TAI - n, where n= complete seconds that can be altered on 1st January or 1st June of any given year (leap seconds).

4.21.3 GPS time

General GPS system time is specified by a week number and the number of seconds with in that week . The star date was Sunday, 6th January 1980 at 0.00hours (UTC). Each GPS week start in the night from Saturday to Sunday, the continuous time scale being set by the main clock at the Master Control Station. The time difference that arises between GPS and UTC time is constantly being calculated and appended to the navigation message.

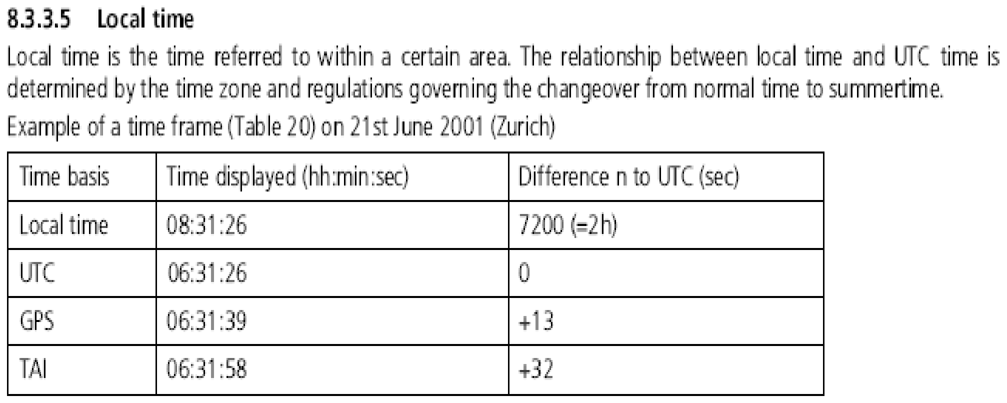
4.21.4 Satellite time

Because of constant, irregular frequency errors in the atomic clocks on board the GPS satellites, individual satellite time is at variance with GPS system time. The satellite clocks are monitored by the control station and any apparent time difference relayed to Earth. Any time differences must be taken into account when Conducting local GPS measurements

4.21.5 Local time

Local time is the time referred to within a certain area. The relationship between local time and UTC time is determined by the time zone and regulations governing the change over from normal time to summer time

Example of a timeframe(Table20)on21stJune2001(Zurich)



8.3.3.5 Local time

Local time is the time referred to within a certain area. The relationship between local time and UTC time is determined by the time zone and regulations governing the changeover from normal time to summertime.

Example of a time frame (Table 20) on 21st June 2001 (Zurich)

Time basis	Time displayed (hh:min:sec)	Difference n to UTC (sec)
Local time	08:31:26	7200 (=2h)
UTC	06:31:26	0
GPS	06:31:39	+13
TAI	06:31:58	+32

Table 11: Time systems

The interrelationship of time systems (valid for 2001):

TAI-UTC=+32sec

GPS-UTC=+13sec

TAI–GPS=+19sec

The purpose of the RS-232 interface is mainly

• to link computers to each other(mostly bi-directional)

• To control serial printers

• To connect PCs to external equipment, such as GSM modems, GPS receivers, etc.

The serial ports in PCs are designed for a synchronous transfer. Persons engaged in transmitting and receiving operations must adhere to a compatible transfer protocol, i.e. an agreement on how data is to be transferred .Both partners must work with the same interface configuration, and this will affect the rate transferum asured In baud. The baud rate is the number of bits per second to be transferred.

Data Is transmitted in inverted logic on the TxD and RxD lines. T stands for transmitter and R for receiver.in accordance with standards, the levels are:

• Logical0=positive voltage, transmit t mode:+5..+15V,receivemode:+3..+15V

• Logical1=negative voltage, transmit mode:-5..-15V,receivemode-3..-15V

The following figure (Figure) illustrates the difference betweenTTL and RS-232levels.Levelinversion can clearly be seen.

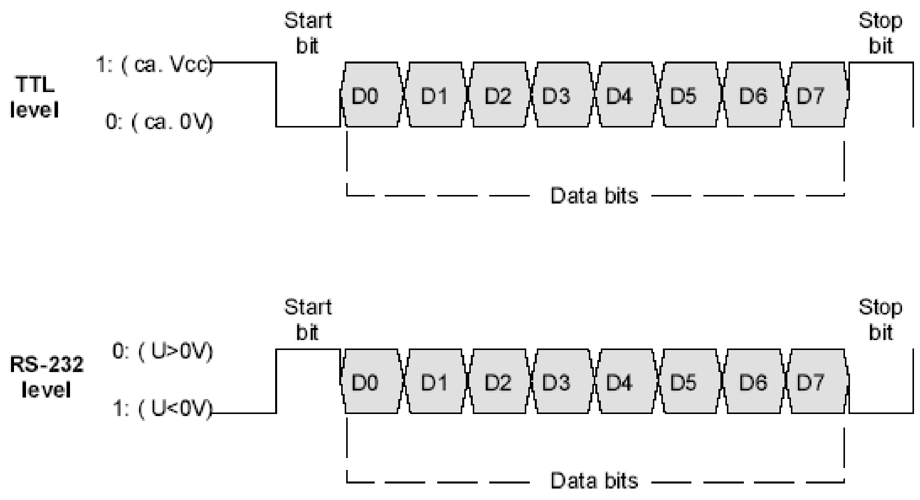


Fig 4.13Difference between TTL and RS-232 levels converting the TTL level to RS-232

Many GPS receivers and GPS modules only make serial NMEA and proprietary data available using TTL levels (approx.0Vorapprox.Vcc=+3.3Vor+5V).It is not always possible to evaluate this data directly through a PC, as a PC input requires RS232level values. As a circuits needed to carry out the necessary level adjustment, the industry has developed integrated circuits Specifically designed to deal with conversion between the two level ranges, to undertake signal inversion ,and to accommodate the necessary equipment to generate negative supply voltage(by means of built-incharge pumps).

A complete bidirectional level converter that uses a "MaximMAX3221"[xxiv] is illustrated on the following Circuit diagram (Figure_50) .The circuit has an operational voltage of 3V...5V and is protected against voltage peaks (ESD)of±15kV.The function of the C1..C4 capacitors is to increase or invert the voltage.

CHAPTER 5

LIQUID CRYSTAL DISPLAY

5.1 Introduction

LCD stands for liquid crystal display. Character and graphical LCD's are most common among hobbyist and DIY electronic circuit/project makers. Since their interface serial/parallel pins are defined so it's easy to interface them with many microcontrollers. Many products we see in our daily life have LCD's with them. They are used to show status of the product or provide interface for inputting or selecting some process. Washing machine, microwave, air conditioners and mat cleaners are few examples of products that have character or graphical LCD's installed in them. In this tutorial i am going to discuss about the character LCD's. How they work? Their pin out and initialization commands etc.

Character LCD's come in many sizes 8x1, 8x2, 10x2, 16x1, 16x2, 16x4, 20x2, 20x4, 24x2, 30x2, 32x2, 40x2 etc. Many multinational companies like Philips, Hitachi, and Panasonic make their own custom type of character LCD's to be used in their products. All character LCD's performs the same functions (display characters numbers special characters, asci characters etc.).Their programming is also same and they all have same 14 pins (0-13) or 16 pins (0 to 15). In an mxn LCD. M denotes number of coulombs and n represents number of rows. Like if the LCD is denoted by 16x2 it means it has 16 coulombs and 2 rows. Few examples are given below. 16x2, 8x1 and 8x2 LCD are shown in the picture below. Note the difference in the rows and coulombs.

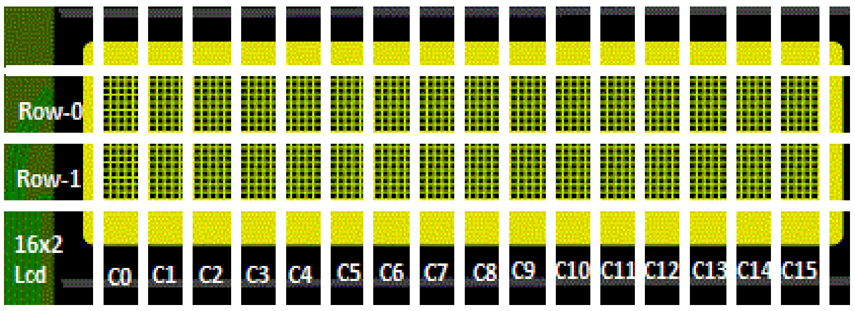


Fig 5.1:LCD

On a character LCD a character is generated in a matrix of 5x8 or 5x7. Where 5 represents number of coulombs and 7/8 represent number of rows. Maximum size of the matrix is 5x8. You cannot display character greater then 5x8 dimension matrix. Normally we display a character in 5x7 matrixes and left the 8th row for the cursor. If we use the 8th row of the matrix for the character display, then there will be no room for cursor. The picture on the right side shows the 5x8 dot matrix pixels arrangement. To display character greater than this dimension you have to switch to graphical LCD's.

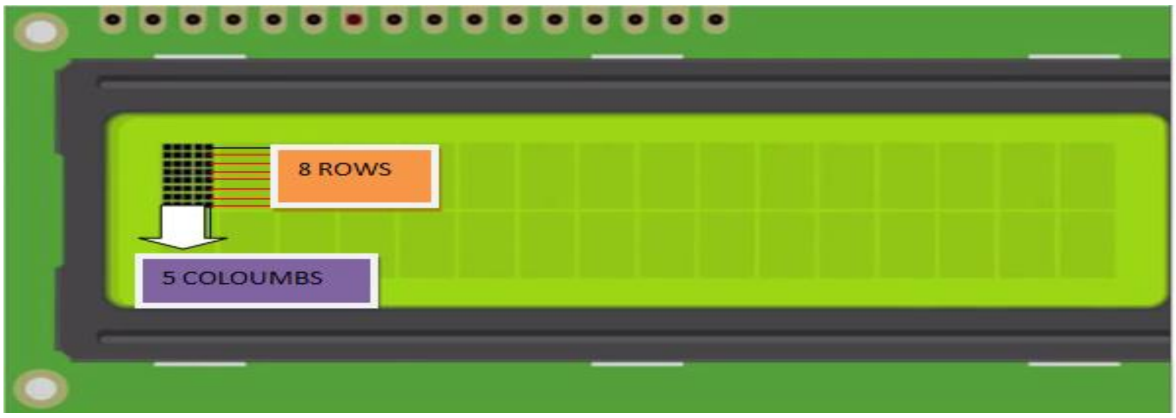
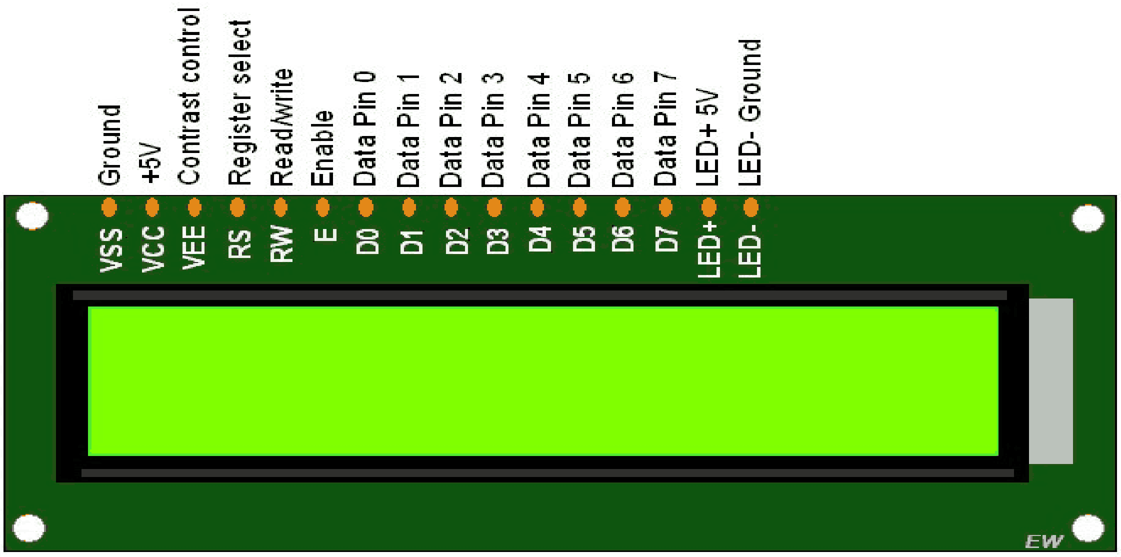


Fig 5.2: Innerview of LCD



All character LCD's have

- Eight (8) data pins D0-D7
- Vcc (Apply +5 volt here)
- Gnd (Ground this pin)
- Rc (Register select)
- Rw (read - write)
- En (Enable)
- V0 (Set LCD contrast)

The picture on the left side shows the pin out of the character LCD. Almost all the character LCD's are composed of the same pin out. LCD's with total pin count equal to 14 does not have back light control option. They might have back light always on or does not have a back light. 16 total pin count LCD's have 2 extra A and K pins. A means anode

Register select selects the HD44780 controller registers. It switches between Command and data register.

- Command Register
- Data Register

a) Command Register

When we send commands to LCD these commands go to Command register and are processed there. Commands with their full description are given in the picture below. When Rs=0 command register is selected.

b) Data Register When we send Data to LCD it goes to data register and is processed there. When Rs=1 data register is selected.

5.3 RW (Read - Write)

Rw pin is used to read and write data to HD44780 data and command registers. When Rw=1 we can read data from LCD. When Rw=0 we can write to LCD.

5.4 En (Enable signal)

When we select the register Rs(Command and Data) and set Rw(read - write) and placed the raw value on 8-data lines, now it's time to execute the instruction. By instruction i mean the 8-bit data or 8-bit command present on Data lines of LCD. For sending the final data/command present on the data lines we use this enable pin. Usually it remains en=0 and when we want to execute the instruction we make it high en=1 for some mills seconds. After this we again make it ground en

5.5 V0 (Set LCD contrast)

To set LCD display sharpness use this pin. Best way is to use variable resistor such as potentiometer a variable current makes the character contrast sharp. Connect the output of the potentiometer to this pin. Rotate the potentiometer knob forward and backward to adjust the LCD contrast.

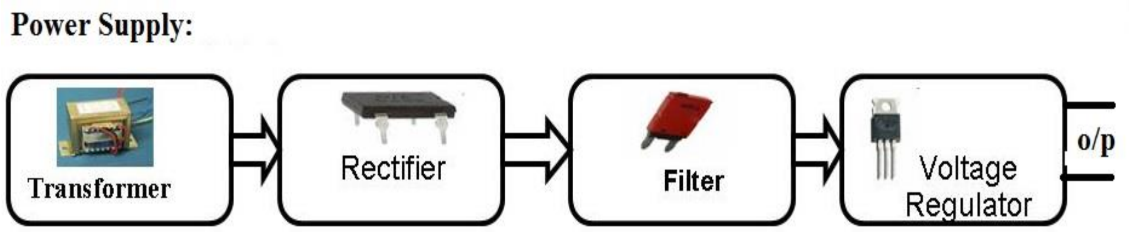
NOTE: we cannot send an integer, float, long, double type data to LCD because LCD is designed to display a character only. Only the characters that are supported by the HD44780 controller. See the HD44780 data sheet to find out what characters can we display on LCD. The 8 data pins on led carries only Ascii 8-bit code of the character to LCD. However we can convert our data in character type array and send one by one our data to LCD. Data can be sent using LCD in 8-bit or 4-bit mode. If 4-bit mode is used, two nibbles of data (First high four bits and then low four bits) are sent to complete a full eight-bit transfer. 8-bit mode is best used when speed is required in an application and at least ten I/O pins are available. 4-bit mode requires a minimum of seven bits. In 4-bit mode, only the top 4 data pins (4-7) are used.

CHAPTER 6

POWER SUPPLY

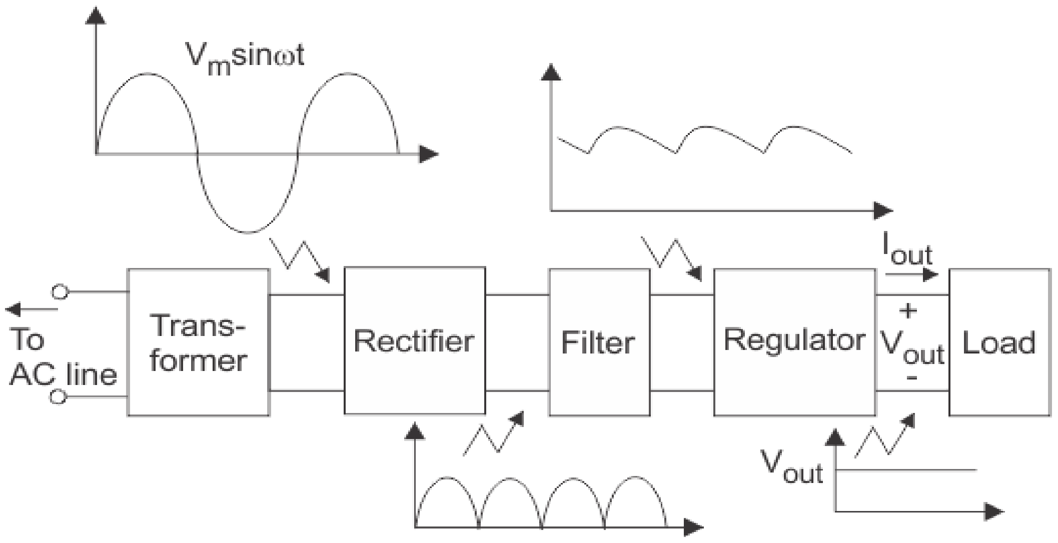
6.1 Regulated power supply

A regulated power su**pply** 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.



A regulated DC power supply is also called as a linear power supply; it is an embedded circuit and consists of various blocks. The regulated power supply will accept an AC input and give a constant DC output

Fig6.1 the block diagram of a typical regulated DC power supply.



Components of typical linear power supply

The basic building blocks of a regulated DC power supply are as follows:

1. A step down transformer
2. A rectifier
3. A DC filter
4. A regulator

6.2 Operation of Regulated Power Supply:

6.2.1 Step Down Transformer:

A step down transformer will step down the voltage from the ac mains to the required voltage level. The turn's ratio of the transformer is so adjusted such as to obtain the required voltage value. The output of the transformer is given as an input to the rectifier circuit.

6.2.2 Rectification:

Rectifier is an electronic circuit consisting of diodes which carries out the rectification process. Rectification is the process of converting an alternating voltage or current into corresponding direct (DC) quantity. The input to a rectifier is ac whereas its output is unidirectional pulsating DC. Usually a full wave rectifier or a bridge rectifier is used to rectify both the half cycles of the ac supply (full wave rectification). Figure below shows a full wave bridge rectifier.

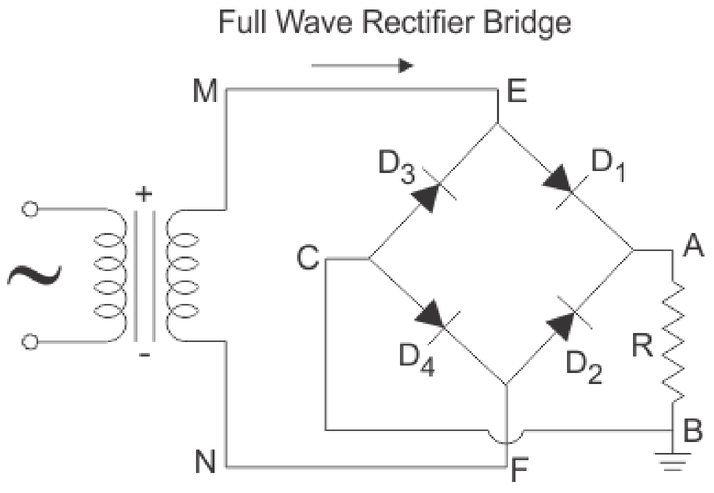
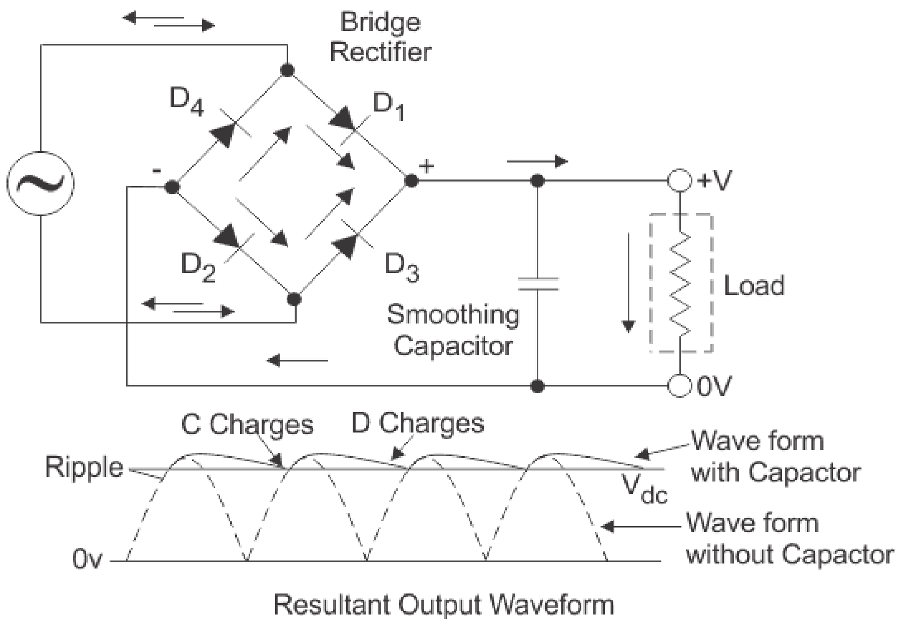


Fig 6.2.2 Full wave bridge rectifier

A bridge rectifier consists of four p-n junction diodes connected in the above shown manner. In the positive half cycle of the supply the voltage induced across the secondary of the electrical transformer i.e. VMN is positive. Therefore point E is positive with respect to F. Hence, diodes D_3 and D_2 are reversed biased and diodes D_1 and D_4 are forward biased. The diode D_3 and D_2 will act as open switches (practically there is some voltage drop) and diodes D_1 andD_4 will act as closed switches and will start conducting.

6.2.3 DC Filtration:

The rectified voltage from the rectifier is a pulsating DC voltage having very high ripple content. But this is not we want, we want a pure ripple free DC waveform. Hence a filter is used. Different types of filters are used such as capacitor filter, LC filter, Choke input filter, π type filter. Figure below shows a capacitor filter connected along the output of the rectifier and the resultant output waveform.



6.2.4 Regulation:

This is the last block in a regulated DC power supply. The output voltage or current will change or fluctuate when there is change in the input from ac mains or due to change in load current at the output of the regulated power supply or due to other factors like temperature changes.

CHAPTER 7

Source code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(2,3,4,5,6,7);

const int button=A0;

const int buz =A5;

const int camera=A1;

#include <Wire.h>

#include <SPI.h>

//https://www.google.com/maps/place/Chalapathi+Institute+of+Technology/@16.46394
96,80.4005482,16z/data=!4m5!3m4!1s0x0:0xecc6c515c0f6d07!8m2!3d16.4636512!4d
80.4044535

#include<SoftwareSerial.h>

//SoftwareSerial Serial1(8,9); //make RX arduino line is pin 2, make TX arduino line is
pin 3.

SoftwareSerial gps(10,11);

int cnt=0,hb=0,ht=0;

int temp=0;

int val;

int i=0,k=0;

int  gps_status=0;

float latitude= 18.6211506;   //18.71540;  18.445898, 79.155876

float logitude= 79.3798186;   //78.13360;   18.620715,79.3800986

//float latitude=0;

//float logitude=0;

String Speed="";
String gpsString="";
char *test="$GPRMC";
String info="HAI";
void setup ()
{
  lcd.begin(16,2);
  lcd.clear();
  Serial.begin(9600);
  pinMode(button,INPUT);
  pinMode(camera,OUTPUT);
  pinMode(buz,OUTPUT);
  lcd.print("VEHICLE ACCIDENT");
  lcd.setCursor(0,1);
  lcd.print("DEVICE GSM&GPS");
  delay(2000);
  digitalWrite(buz,HIGH);
  lcd.clear();
 lcd.print("Initializing");
  lcd.setCursor(0,1);
  lcd.print("Please Wait...");
  delay(1000);
  lcd.clear();lcd.print("AT");Serial.print("AT\r\n");delay(1000);
  lcd.clear();lcd.print("ATE0");Serial.print("ATE0\r\n");delay(1000);
  lcd.clear();lcd.print("AT+CMGF=1");Serial.print("AT+CMGF=1\r\n");delay(1000);
lcd.clear();lcd.print("AT+CNMI=1,2,0,0");Serial.print("AT+CNMI=1,2,0,0\r\n");delay(
1000);
```

```
lcd.clear();
  lcd.print("Waiting For GPS");
  lcd.setCursor(0,1);
  lcd.print("     Signal    ");
  delay(2000);
  gps.begin(9600);
  //get_gps();
  show_coordinate();
  delay(2000);
  lcd.clear();
  lcd.print("GPS is Ready");
  delay(1000);
  lcd.clear();
  lcd.print("System Ready");
  digitalWrite(button,HIGH);
  lcd.clear();
  //digitalWrite(sw,HIGH);
}
void loop ()
{
unsigned int sw =digitalRead(button);
lcd.setCursor(0,0);
 lcd.print("ACCIDENT DETECT");
 lcd.setCursor(0,1);
 lcd.print("DEVICE ");
```

```
  digitalWrite(camera,LOW);
  delay(500);
if(sw==0)
{
  //while(1)
  {
  lcd.clear();
  digitalWrite(buz,LOW);
  lcd.print("PLEASE HELP");
  //digitalWrite(camera,LOW);
  delay(500);
  digitalWrite(camera,HIGH);
  delay(500);
  Send();
  digitalWrite(buz,HIGH);
  delay(1000);delay(1000);delay(1000);delay(1000);
  delay(1000);delay(1000);delay(1000);delay(1000);
  delay(1000);
  }
}


}


void gpsEvent()
{
  gpsString="";
  while(1)
  {
   while (gps.available()>0)           //Serial incoming data from GPS
   {
    char inChar = (char)gps.read();
     gpsString+= inChar;                    //store incoming data from GPS to temparary string
str[]
     i++;
    // Serial.print(inChar);
     if (i < 7)
     {
      if(gpsString[i-1] != test[i-1])        //check for right string
      {
        i=0;
        gpsString="";
      }
     }
    if(inChar=='\r')
    {
     if(i>60)
     {
       gps_status=1;
       break;
     }
     else
     {
       i=0;
     }
    }
  }
   if(gps_status)
    break;
  }
}
void get_gps()
{
  lcd.clear();
  lcd.print("Getting GPS Data");
  lcd.setCursor(0,1);
  lcd.print("Please Wait.....");
   gps_status=0;
   int x=0;
   while(gps_status==0)
   {
    gpsEvent();
    int str_lenth=i;
    coordinate2dec();
    i=0;x=0;
    str_lenth=0;
   }
```

```
}
void show_coordinate()
{
   lcd.clear();
   lcd.print("Lat:");
   lcd.print(latitude);
   lcd.setCursor(0,1);
   lcd.print("Log:");
   lcd.print(logitude);
 /*  Serial.print("Latitude:");
   Serial.println(latitude);
   Serial.print("Longitude:");
   Serial.println(logitude);
   Serial.print("Speed(in knots)=");
   Serial.println(Speed);
   delay(2000);
   */
   lcd.clear();
   lcd.print("Speed(Knots):");
   lcd.setCursor(0,1);
   lcd.print(Speed);
}
void coordinate2dec()
{
  String lat_degree="";
  for(i=19;i<=20;i++)
    lat_degree+=gpsString[i];
    Serial.println(lat_degree);
String lat_minut="";
   for(i=21;i<=30;i++)
    lat_minut+=gpsString[i];
    Serial.println(lat_minut);
String log_degree="";
  for(i=32;i<=34;i++)
    log_degree+=gpsString[i];
    Serial.println(log_degree);
String log_minut="";
  for(i=35;i<=44;i++)
    log_minut+=gpsString[i];
    Serial.println(log_minut);
  Speed="";
  for(i=45;i<48;i++)          //extract longitude from string
    Speed+=gpsString[i];

   float minut= lat_minut.toFloat();
   minut=minut/60;
   float degree=lat_degree.toFloat();
   latitude=degree+minut;
   Serial.println(latitude);
```

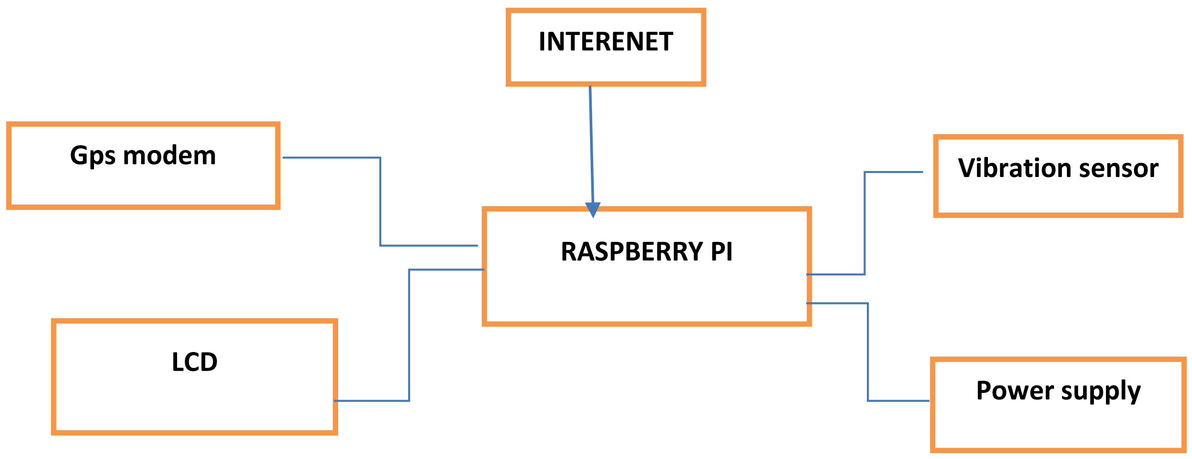
```
    minut= log_minut.toFloat();
    minut=minut/60;
    degree=log_degree.toFloat();
    logitude=degree+minut;
    Serial.println(logitude);
}
void Send()
{
  Serial.println("AT");
  delay(500);
  serialPrint();
  Serial.println("AT+CMGF=1");
  delay(500);
  serialPrint();
  Serial.print("AT+CMGS=");
  Serial.print("");
  Serial.print("8501803670");    //mobile no. for SMS alert
  Serial.println("");
  delay(1000);
  serialPrint();
  Serial.print("Latitude:");
  Serial.println(latitude,6);
  delay(500);
  serialPrint();
  Serial.print(" longitude:");
```

```
   Serial.println(logitude,6);
   delay(500);
   serialPrint();
   delay(500);
   serialPrint();
   Serial.print(info);
   Serial.println("problem occured at the location of");
   Serial.print("http://maps.google.com/maps?&z=15&mrt=yp&t=k&q=");
   Serial.print(latitude,6);
   Serial.print("+");              //28.612953, 77.231545   //28.612953,77.2293563
   Serial.print(logitude,6);
   Serial.write(26);
   delay(2000);
   serialPrint();
  // lcd.clear();
}
void serialPrint()
{/*
  while(Serial1.available()>0)
  {
    Serial.print(Serial1.read());
  }*/
}
```

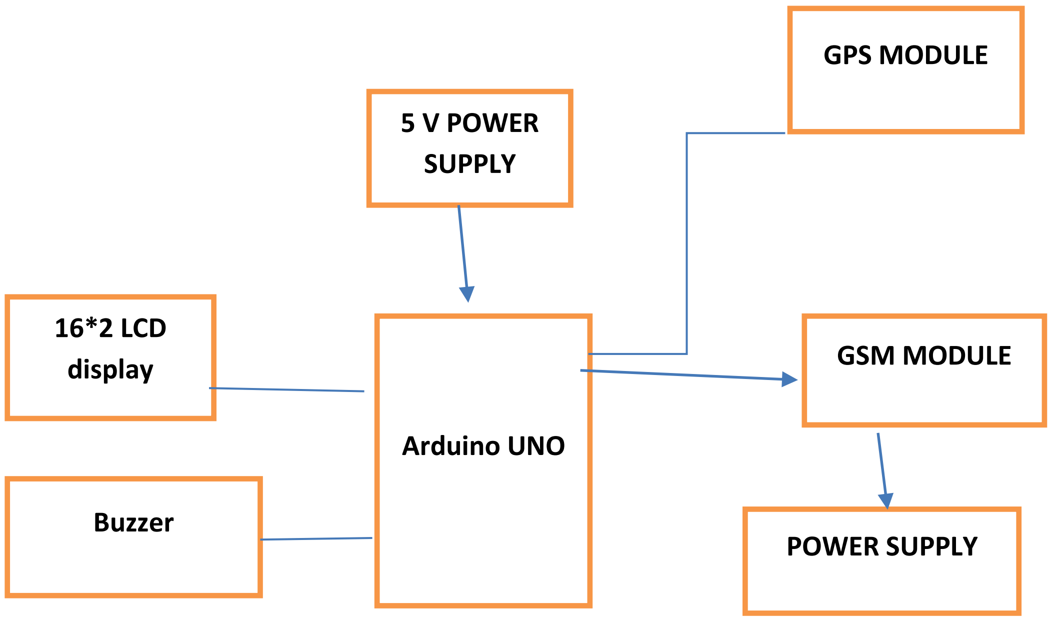
CHAPTER 8

BLOCK DIAGRAM

8.1 Block diagram of existing system

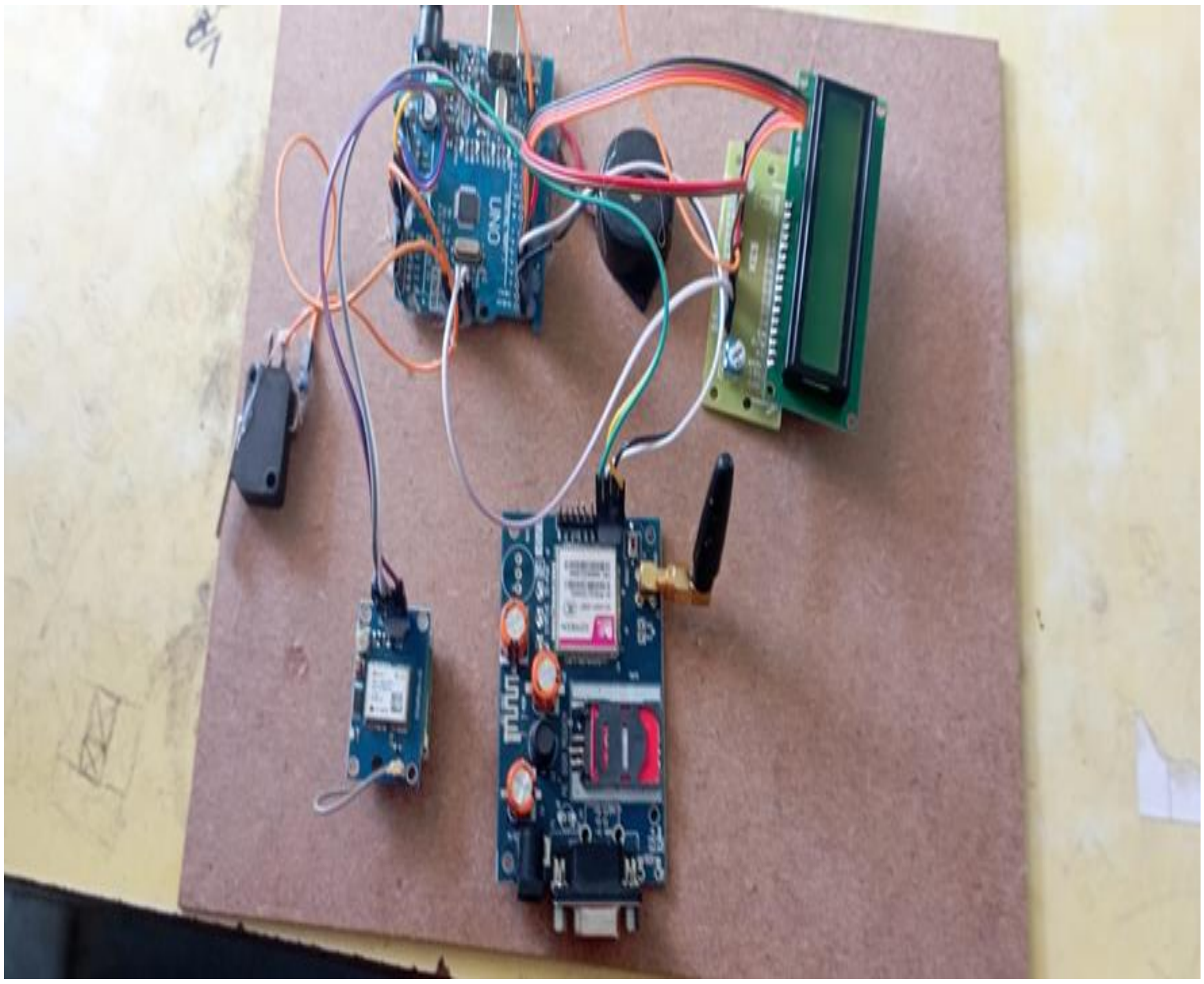


8.2 Block diagram of proposed system



CHAPTER 9

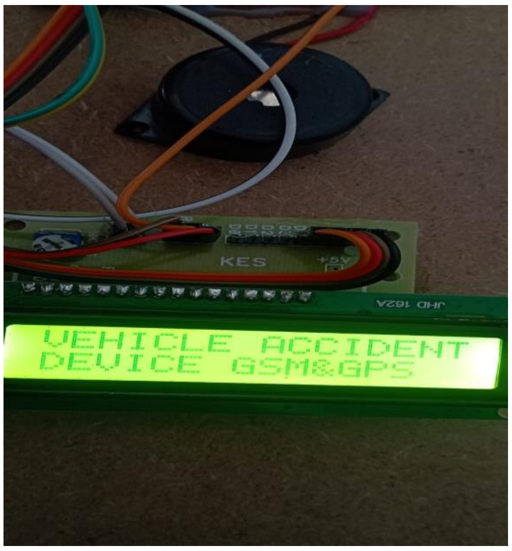
HARDWARE IMPEMENTATION

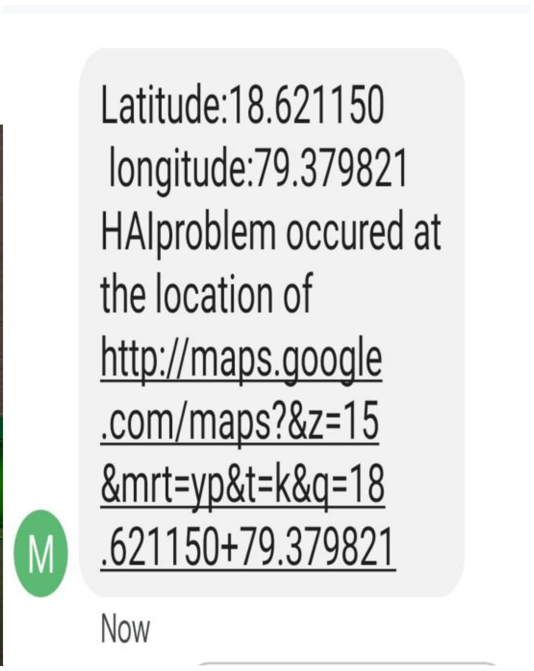


CHAPTER 10

RESULT AND ANALYSIS

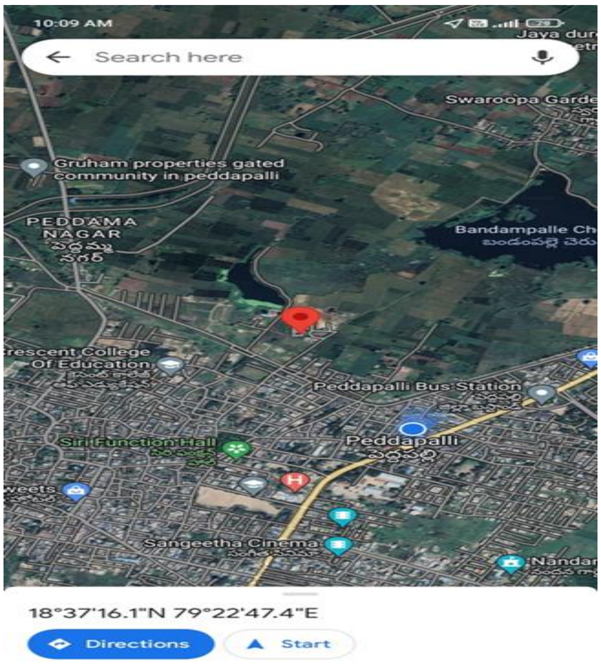
RESULT:





Latitude:18.621150
longitude:79.379821
HAIproblem occured at the location of http://maps.google.com/maps?&z=15&mrt=yp&t=k&q=18.621150+79.379821

Now



18°37'16.1"N 79°22'47.4"E

CHAPTER 11

Advantages and dis advantages

Disadvantages of existing system:

- Costlier.
- Sending data not secure.
- This system is not applicable for poor network connection places.

Advantages of proposed system:

- Cost is less
- Monitor all hazards and threats in both network coverage and no network areas.
- Wireless monitoring and user friendly.
- Fast recovery and quick process.

CHAPTER 12

APPLICATIONS

- ❖ Automotive and transport vehicles.
- ❖ Security, remote monitoring and transportation and logistics.
- ❖ This system also can be interfaced with vehicle alerting system.
- ❖ Smart car

CHAPTER 13

CONCLUSION

The proposed system uses the IoT for vehicle accident detection and alarming the authorities regarding accidents, vehicle tracking using GPS Modem. In this project we have designed IoT based vehicle accident detection and tracking system using GPS Modem. Hence IoT can revolutionize the way the system interact and respond for the variety of applications especially in case of traffic control**.**

CHAPTER 14

FUTURE SCOPE

This project work can be used to reach the desired location by avoid obstacles in planets like mars to detect metals. The proposed program deals with detecting incidents and warning paramedics to reach the specific location. This can be extended through providing the victim with medication at the spot of the accident. We can also avoid accidents by increasing the technology and using warning systems that could really stop the vehicle to conquer them.

Literature survey

- ❖ Many researchers carried out their studies on accident detection system. Aishwarya S.R explained an IoT based vehicle accident prevention and tracking system for night drivers

The 8051 Micro controller and Embedded Systems

Muhammad Ali Mazidi
Janice Gillispie Mazidi

The 8051 Micro controller Architecture, Programming & Applications

Kenneth J. Ayala

Fundamentals of Micro processors and Micro computers

B. Ram

Micro processor Architecture, Programming & Applications

Ramesh S. Gaonkar

Electronic Components

D.V. Prasad

References on the Web:

www.national.com
www.atmel.com
www.microsoftsearch.com
www.geocities.com