AUTOMATED ACCIDENT DETECTION AND INFORMATION SYSTEM [AADIS]

A PROJECT REPORT

Submitted by

R.RAJESH ANAND	**2012505519**
N.R.NANJAYAN	**2012505546**
S.MANOJ	**2012505545**

For Centre for Technology Development and Transfer project

ELECTRONICS AND INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY:: CHENNAI 600 044

OCTOBER 2015

MADRAS INSTITUTE OF TECHNOLOGY CAMPUS,

ANNA UNIVERSITY: CHENNAI-600 044

BONAFIDE CERTIFICATE

Certified that this project report " AUTOMATED ACCIDENT DETECTION AND INFORMATION SYSTEM [AADIS] " is the bonafide work of NANJAYAN.N.R (2012505546), RAJESH ANAND.R(2012505519), S.MANOJ (2012505545) who carried out the project work under my supervision.

SIGNATURE OF HOD	SIGNATURE OF GUIDE
Dr.J.PRAKASH	**Ms.M.MYTHILI**
Head Of The Department	**Assistant Professor**
Dept. of Instrumentation Engineering	Dept. of Instrumentation Engineering
Madras Institute of Technology,	Madras Institute of Technology,
Anna University,	Anna University,
Chennai – 600 044	Chennai – 600 044

ACKNOWLEDGEMENT

We wish to place on record, our deep gratitude and appreciation to our dean and Professor **Dr.A.Rajadurai** for providing the necessary infrastructure to carry out the project.

We are thankful to our Head of the Department, **Dr.J.Prakash**, who has encouraged and motivated us in our endeavors.

We are extremely grateful to our project guide **Ms.M.Mythili**, for his timely and thoughtful guidance without whom this project could not have been realized.

We would also like to our professor **Dr.K.Arunachalam**, for helping us through the course of the project.

We owe a debt of gratitude to all the staff members of the Department of Instrumentation Engineering and our friends who constantly offered constructive suggestions for improvement in various phases of the project.

NANJAYAN.N.R

RAJESH ANAND.R

MANOJ.S

ABSTRACT

The aim of our Centre for Technology and Development Transfer Project is to provide a quick short message service to rescue team or to an individual number to seek immediate medical attention to the victims. The Rapid growth of technology and infrastructure has made our lives easier. The advent of technology has also increased the traffic hazards and the road accidents take place frequently which causes huge loss of life and property because of the poor emergency facilities. Our project will provide an optimum solution to this draw back. An accelerometer sensor is used in a car so that dangerous accident can be detected. The accelerometer sensor detects the vibration of the vehicle during and after a crash. With signals from an accelerometer, a severe accident can be recognized. According to this project when a vehicle meets with an accident immediately Vibration sensor will detect the signal and sends it to ARM controller. Microcontroller sends the alert message through the GSM MODEM including the location to police control room or a rescue team or to a personal number. So the police can immediately trace the location through the GPS MODEM, after receiving the information. Then after conforming the location, necessary action will be taken. If the person meets with a small accident or if there is no serious threat to anyone`s life, then the alert message can be terminated by the driver by a switch. It is provided in order to avoid wasting the valuable time of the medical rescue team.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The high demand of automobiles has also increased the traffic hazards and the road accidents. This is because of the lack of best emergency facilities available in our country. An automatic alarm device for vehicle accidents can be made. This design is a system which can detect accidents in significantly less time and sends the basic information to first aid center within a few seconds covering geographical coordinates, the time and angle in which a vehicle accident had occurred. This alert message is sent to the rescue team in a short time, which will help in saving the valuable lives.

A switch is also provided in order to terminate the sending of a message in rare case where there is no casualty, this can save the precious time of the medical rescue team.

When the accident occurs the alert message is sent automatically to the rescue team and to the police station. The message is sent through the GSM module and the location of the accident is detected with the help of the GPS module. The accident can be detected precisely with the help of vibration sensor. This application provides the optimum solution to poor emergency facilities provided to the roads accidents in the most feasible way .

1.1 NEED OF THE PROJECT

Over 1,37,000 people were killed in road accidents in 2013 alone, that is more than the number of people killed in all our wars put together. There is one death every four minutes due to a road accident in India. 1214 road crashes occur every day in India. One serious road accident in the country occurs every minute and 16 die on Indian roads every hour. Two people die every hour in Uttar Pradesh – State with maximum number of road crash deaths. Tamil Nadu is the state with the maximum number of road crash injuries. Chennai is the 2nd highest city to have highest number of road accident deaths. Hence, this statistics motivated us to provide a quick and reliable system to provide emergency message to rescue team. And also , it takes several hours for friends and family of the victim to know about the incident. So, we provided an emergency number of the individual to generate a short message service .

1.2 OBJECTIVES OF THE PROJECT

The specific objective of this project are

1.2.1 PRIMARY OBJECTIVE

- To provide immediate medical attention to the injured persons by generating a short message service.

1.2.2 SECONDARY OBJECTIVE

- To notify the kith and kin of the victims about incident by having an emergency number prior.

1.3 ORGANIZATION OF THE THESIS

- **Chapter 1:** This chapter deals with importance of Accident information system. It introduces the domain of the problem considered and also the objective of the project. It outlines the existing systems, their advantages and disadvantages and henceforth the motivation of the project work. This chapter deals with the Budget organization of this project in the span of three months. It also includes the scope of the project.

- **Chapter 2** This chapter deals with the methodology with which the project has been proceeded. It also includes the schematic representation of the project.

- **Chapter 3:** It includes the best suited components for Vibration measurement . It provides information about the ARM processor.

- **Chapter 4:** This chapter deals with the results of the proposed technique. The performance of the project has been included.

- **Chapter 5:** This chapter deals with the conclusion and scope for future work of the project.

1.4 LITERATURE REVIEW

1.4.1 TYPES OF ACCELERATION SENSOR

GYROSCOPE

A vibrating structure gyroscope, standardized by IEEE as Coriolis vibratory gyroscope (CVG),[1] is a wide group of gyroscope using solid-state resonators of different shapes that functions much like the halteres of an insect.

The underlying physical principle is that a vibrating object tends to continue vibrating in the same plane as its support rotates. In the engineering literature, this type of device is also known as a Coriolis vibratory gyro because as the plane of oscillation is rotated, the response detected by the transducer results from the Coriolis term in its equations of motion ("Coriolis force").

Vibrating structure gyroscopes are simpler and cheaper than conventional rotating gyroscopes of similar accuracy. Miniature devices using this principle are a relatively inexpensive type of attitude indicator.

PIEZOELECTRIC ACCELEROMETER

A piezoelectric accelerometer is an accelerometer that employs the piezoelectric effect of certain materials to measure dynamic changes in mechanical variables (e.g., acceleration, vibration, and mechanical shock).

As with all transducers, piezoelectric accelerometers convert one form of energy into another and provide an electrical signal in response to a quantity, property, or condition that is being measured. Using the general sensing method upon which all accelerometers are based, acceleration acts upon a seismic mass that is restrained by a spring or suspended on a cantilever beam, and converts a physical force into an electrical signal. Before the acceleration can be converted into an electrical quantity it must first be converted into either a force or displacement.

SHOCK DETECTOR

A shock detector or impact monitor is a device which indicates whether or not a specified physical shock or impact has occurred. These usually have a binary output and are sometimes called shock overload devices. Shock detectors can be used on shipments of fragile valuable items to indicate whether a potentially damaging drop or impact may have occurred. They are also used in sports helmets to help determine if a dangerous impact may have occurred. By contrast, a shock data logger is a data acquisition system for analysis and recording of shock pulses.

1.4.2 POSITION SENSOR

HALL EFFECT SENSOR

A Hall effect sensor is a transducer that varies its output voltage in response to a magnetic field. Hall effect sensors are used for proximity switching, positioning, speed detection, and current sensing applications.

In its simplest form, the sensor operates as an analog transducer, directly returning a voltage. With a known magnetic field, its distance from the Hall plate can be determined. Using groups of sensors, the relative position of the magnet can be deduced. Frequently, a Hall sensor is combined with circuitry that allows the device to act in a digital (on/off) mode, and may be called a switch in this configuration. Commonly seen in industrial applications such as the pictured pneumatic cylinder, they are also used in consumer equipment; for example some computer printers use them to detect missing paper and open covers. When high reliability is required, they are used in keyboards.

Hall sensors are commonly used to time the speed of wheels and shafts, such as for internal combustion engine ignition timing, tachometers and anti-lock braking systems. They are used in brushless DC electric motors to detect the position of the permanent magnet. In the pictured wheel with two equally spaced magnets, the voltage from the sensor will peak twice for each revolution. This arrangement is commonly used to regulate the speed of disk drives.

ULTRASONIC SENSOR

Ultrasonic transducers are transducers that convert ultrasound waves to electrical signals or vice versa. Those that both transmit and receive may also be called ultrasound transceivers; many ultrasound sensors besides being sensors are indeed transceivers because they can both sense and transmit. These devices work on a principle similar to that of transducers used in radar and sonar systems, which evaluate attributes of a target by interpreting the echoes from radio or sound waves, respectively. Active ultrasonic sensors generate high-frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object. Passive ultrasonic sensors are basically microphones that detect ultrasonic noise that is present under certain conditions, convert it to an electrical signal, and report it to a computer.

1.4.3 ARM PROCESSOR

ARM, originally Acorn RISC Machine, is a family of reduced instruction set computing (RISC) instruction set architectures for computer processors, configured for various environments, developed by British company ARM Holdings.

A RISC-based computer design approach means ARM processors require significantly fewer transistors than typical complex instruction set computing (CISC) x86processors in most personal computers. This approach reduces costs, heat and power use. Such reductions are desirable traits for light, portable, battery-powered devices—including smart phones, laptops, tablet and notepad computers, and other embedded systems. A simpler design facilitates more efficient multi-core CPUs and higher core counts at lower cost, providing improved energy efficiency for servers.

ARM Holdings develops the instruction set and architecture for ARM-based products, but does not manufacture products. The company periodically releases updates to its cores. All cores from ARM Holdings support a 32-bit address space (only pre-ARMv3 chips, as in original Acorn Archimedes, had smaller) and 32-bit arithmetic; the ARMv8-A architecture, announced in October 2011, adds support for a 64-bit address space and 64-bit arithmetic. Instructions for ARM Holdings' cores have 32 bits wide fixed-length instructions, but later versions of the architecture also support a variable-length instruction set that provides both 32 and 16 bits wide instructions for improved code density. Some cores can also provide hardware execution of Java byte codes.

Globally ARM is the most widely used instruction set architecture in terms of quantity produced. The low power consumption of ARM processors has made them very popular: over 50 billion ARM processors have been produced as of 2014, of which 10 billion were produced in 2013 and "ARM-based chips are found in nearly 60 percent of the world's mobile devices". The ARM architecture (32-bit) is the most widely used architecture in mobile devices, and most popular 32-bit one in embedded systems. In 2005, about 98% of all mobile phones sold used at least one ARM processor. According to ARM Holdings, in 2010 alone, producers of chips based on ARM architectures reported shipments of 6.1 billion ARM-based processors, representing 95% of smart phones, 35% of digital televisions and set-top boxes and 10% of mobile computers.

1.5 ORGANISATION

1.5.1 BUDGET ALLOCATION

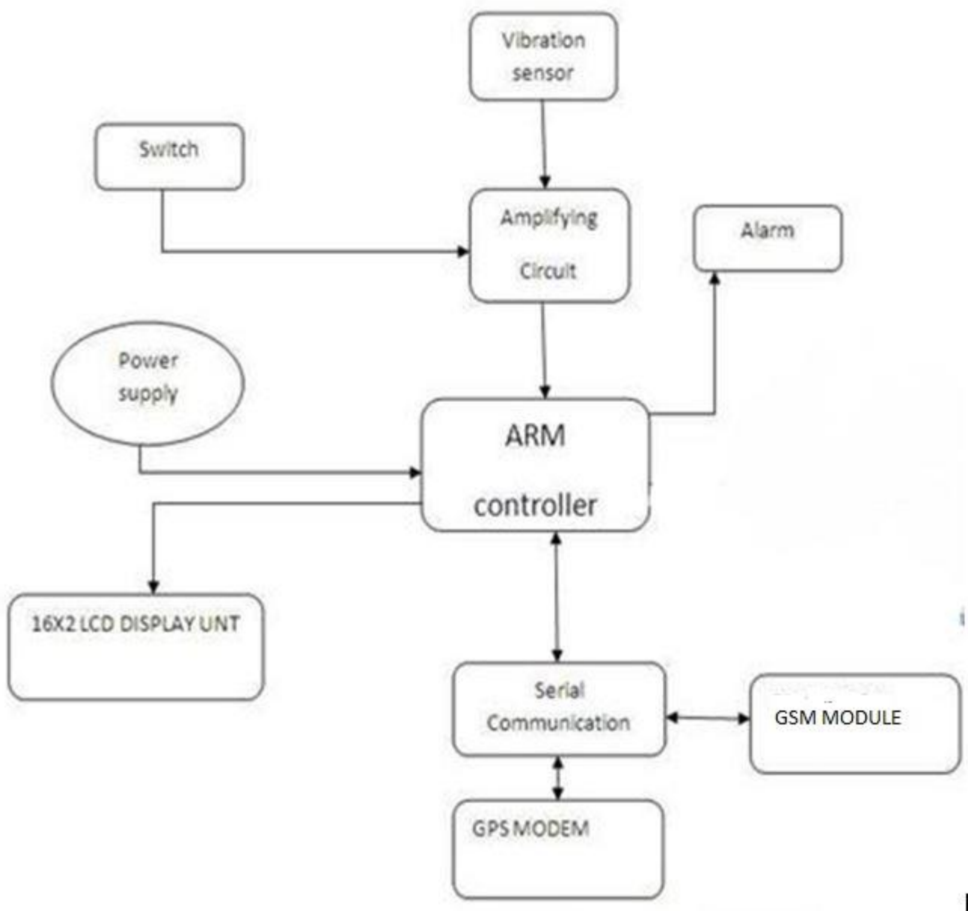
ITEM	COST [INR]
• VIBRATION SENSOR	8000
• ARM PROCESSOR	4000
• GPS MODULE	3000
• GSM MODULE	3000
• LCD INTERFACING CIRCUITS	2000
• ALARM AND AMPLIFYING CIRCUIRS	2000
• TRANSPORT AND MISCELLANEOUS CHARGES	3000
TOTAL	25,000

CHAPTER 2

METHODOLOGY

The main idea is to provide short message after accident to the rescue team. The Sensor ADXL 345 senses the input and sends it to ARM7 Processor. The processor then activates the GPS and GSM module . The GSM SIM 900 module then starts processing the SIM and sends short message with co ordinates given by GPS module. An additional switch is provided to interrupt the message service in case of no injuries.

2.1 BLOCK DIAGRAM



2.1 BLOCK DIAGRAM

2.2 WORKING

The system consists of different modules which are interfaced to the ARM (32 bit) controller. The input power is step down to 12v DC from 230v AC power line by the power supply unit. The main module is the ARM controller which provides high speed processing of the data because of the pipelining technique and ability to be used as a 16 bit controller called Thumb. The main advantage of using this controller is its better performance with high code density.

Whenever the accident occurs the vibrations are sensed by the vibration sensor and these signals are given to the controller through the amplifying circuit. As and when the input is received by the controller, the message is sent to the rescue team with the help of the GSM module. The rescue team reaches the site of the accident with the help of the location given in the message. The location or the geographical coordinates where the vehicle is present are detected by the GPS module.

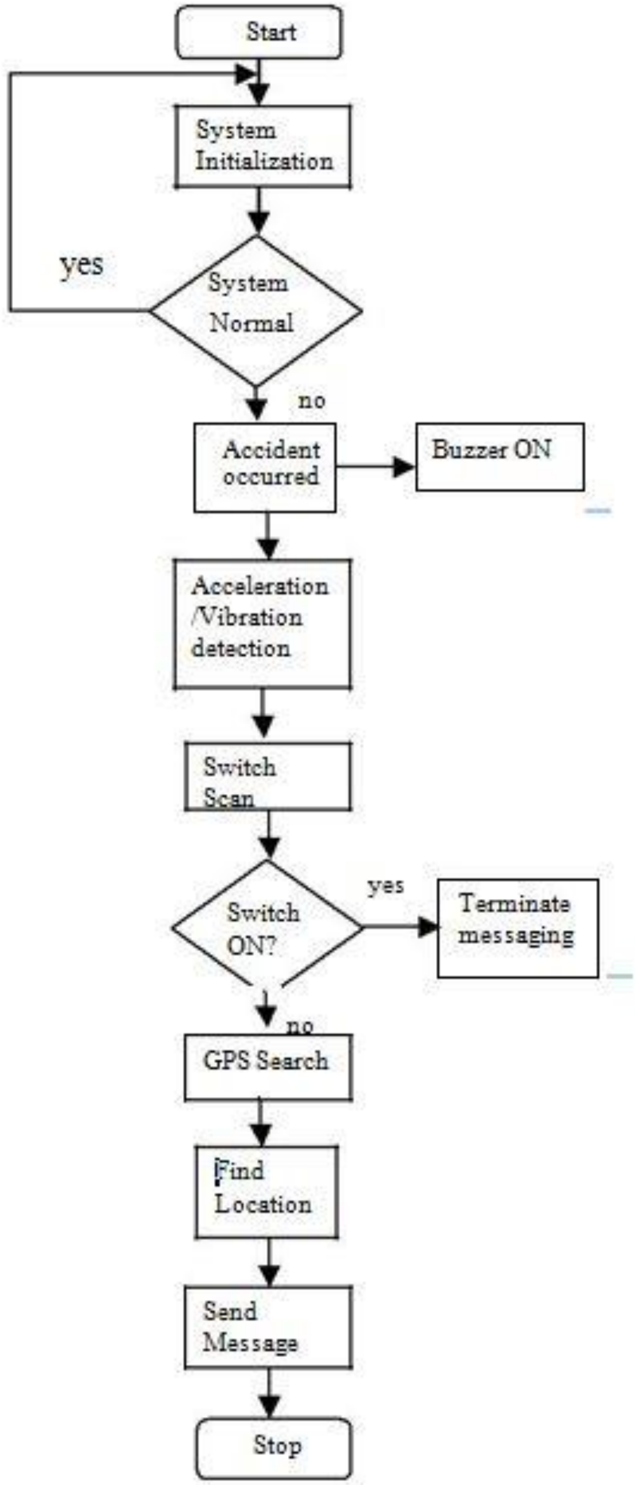
An LCD display is provided to get the display of the tasks carried out. In some conditions where there are no casualties or when there is no need of the medical facility to the person, then the messaging can be terminated with the help of the switch provided in order to avoid wasting the valuable time of the medical rescue team. The GSM and GPS modules are interfaced to the ARM controller using serial communication. All the components are interfaced precisely so that the accident detection and alert message sending are fully automated, so that the warning time is reduced significantly.

2.3 FLOWCHART:

The Flow Chart of the system shows the system is initialized on power ON. When the system is detected to be abnormal, it is confirmed that the accident has occurred. The vibration/acceleration of the vehicle is detected to confirm the cause of the accident.

As soon as the accident is detected, the switch is scanned first; if it is a minor accident then the switch is ON so that messaging is terminated. If it is a major accident, the switch remains OFF and the message is sent automatically to the rescue team after the location is detected by the GPS.

2.2 FLOW CHART



2.2 FLOW CHART

CHAPTER 3

DESIGN AND DEVELOPMENT

3.1 COMPONENT DETAILS

I. ARM LPC 2148

ARM is a family of instruction set architectures for computer processors based on a reduced instruction set computing (RISC) architecture developed by British company ARM Holdings.

A RISC-based computer design approach means ARM processors require significantly fewer transistors than typical processors in average computers. This approach reduces costs, heat and power use. These are desirable traits for light, portable, battery-powered devices—including smart phones, laptops, tablet and notepad computers), and other embedded systems. A simpler design facilitates more efficient multi-core CPUs and higher core counts at lower cost, providing higher processing power and improved energy efficiency for servers and supercomputers.

LPC 2148 is the widely used IC from ARM-7 family. It is manufactured by Philips and it is pre-loaded with many inbuilt peripherals making it more efficient and a reliable option for the beginners as well as high end application developer.

3.1 LPC 2148

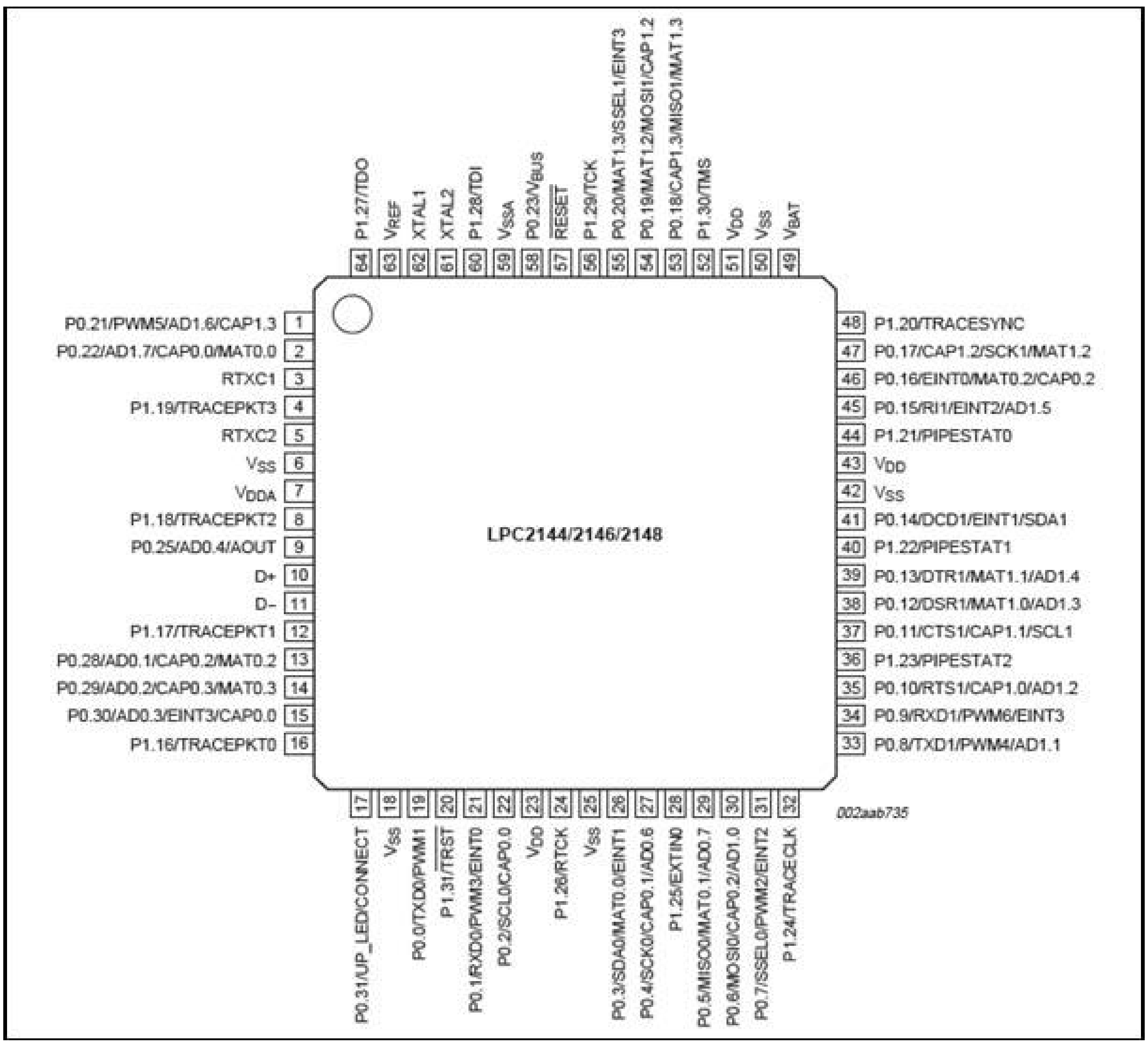
The LPC2148 microcontroller is based on a 32/16 bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combines the microcontroller with embedded high speed flash memory ranging from 32 kB to 512 kB. It is cost effective and reliable. Pipelining is employed in order to simultaneously operate all parts of processing and memory systems. The CPU has two types of instruction sets Arm (32-bit) which gives maximum performance and Thumb (16-bit) which gives maximum code density. The major advantage of ARM is its ability to manipulate 32-bit integers with a single instruction. The main advantage of Thumb is its ability to switch back to ARM which gives high speed to operate fast interrupts and other algorithms. This provides better performance than 16-bit architecture, and better code density than a 32-bit architecture.

FEATURES OF LPC2148.

- 8 to 40 kB of on-chip static RAM and 32 to 512 kB of on-chip flash program memory.128 bit wide interface/accelerator enables high speed 60 MHz operation.
- • In-System/In-Application Programming (ISP/IAP) via on-chip boot-loader software. Single flash sector or full chip erase in 400 ms and programming of 256 bytes in 1ms.
- Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with theon-chip Real Monitor software and high speed tracing of instruction execution.
- USB 2.0 Full Speed compliant Device Controller with 2 kB of endpoint RAM. In addition, the LPC2146/8 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/2 vs. LPC2144/6/8) 10-bit A/D converters provide a total of 6/14analog inputs, with conversion times as low as 2.44 us per channel.
- Single 10-bit D/A converter provides variable analog output.
- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power real-time clock with independent power and dedicated 32 kHz clock input.
- Multiple serial interfaces including two UARTs (16C550), two Fast I2C-bus(400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64.

- Up to nine edge or level sensitive external interrupt pins available.
- On-chip integrated oscillator operates with an external crystal in range from 1 MHz to30 MHz and with an external oscillator up to 50 MHz.
- Power saving modes include Idle and Power-down.
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).
- Single power supply chip with Power-On Reset (POR) and BOD circuits

PIN CONFIGURATION



3.2 PIN CONFIGURATION

LPC2148 need minimum below listed hardware to work properly.

1. Power Supply

2. Crystal Oscillator

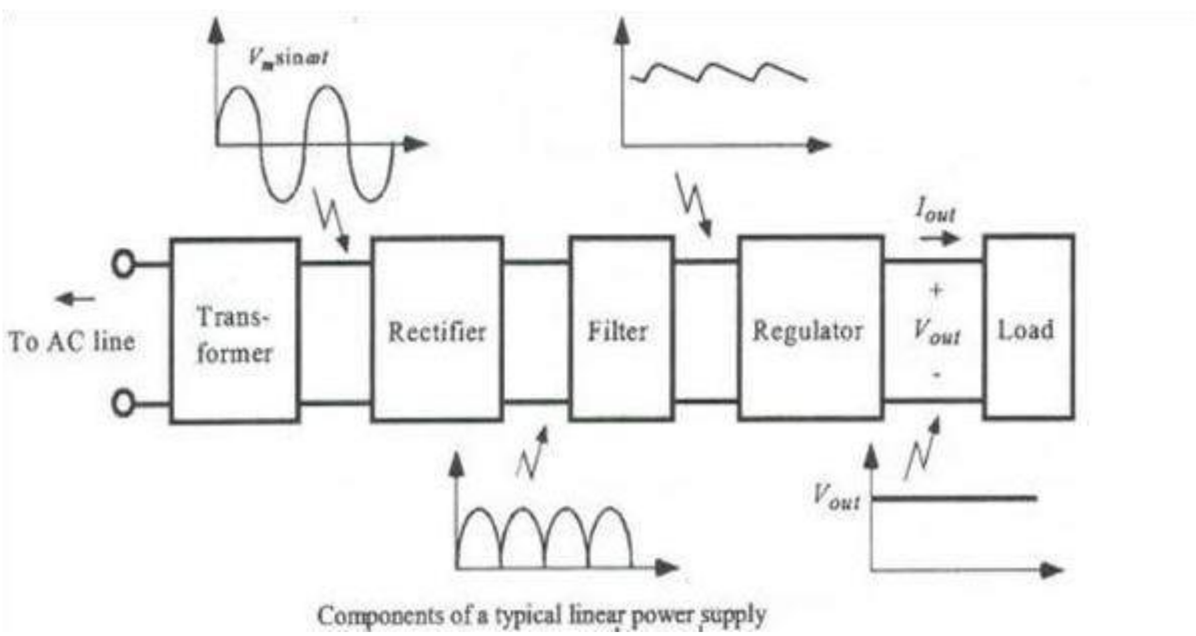
3. Reset Circuit

4. RTC crystal oscillator (This is not necessary if not using RTC).

5. UART

1. Power Supply

LPC2148 works on 3.3 V power supply. LM 117 can be used for generating 3.3 V supply. However, basic peripherals like LCD, ULN 2003 (Motor Driver IC) etc. works on 5V. So AC mains supply is converted into 5V using below mentioned circuit and after that LM 117 is used to convert 5V into 3.3V.



3.3 POWER SUPPLY

Transformer:

It is used to step down 230V AC to 9V AC supply and provides isolation between power grids and circuit.

Rectifier:

It is used to convert AC supply into DC.

Filter:

It is used to reduce ripple factor of DC output available from rectifier end.

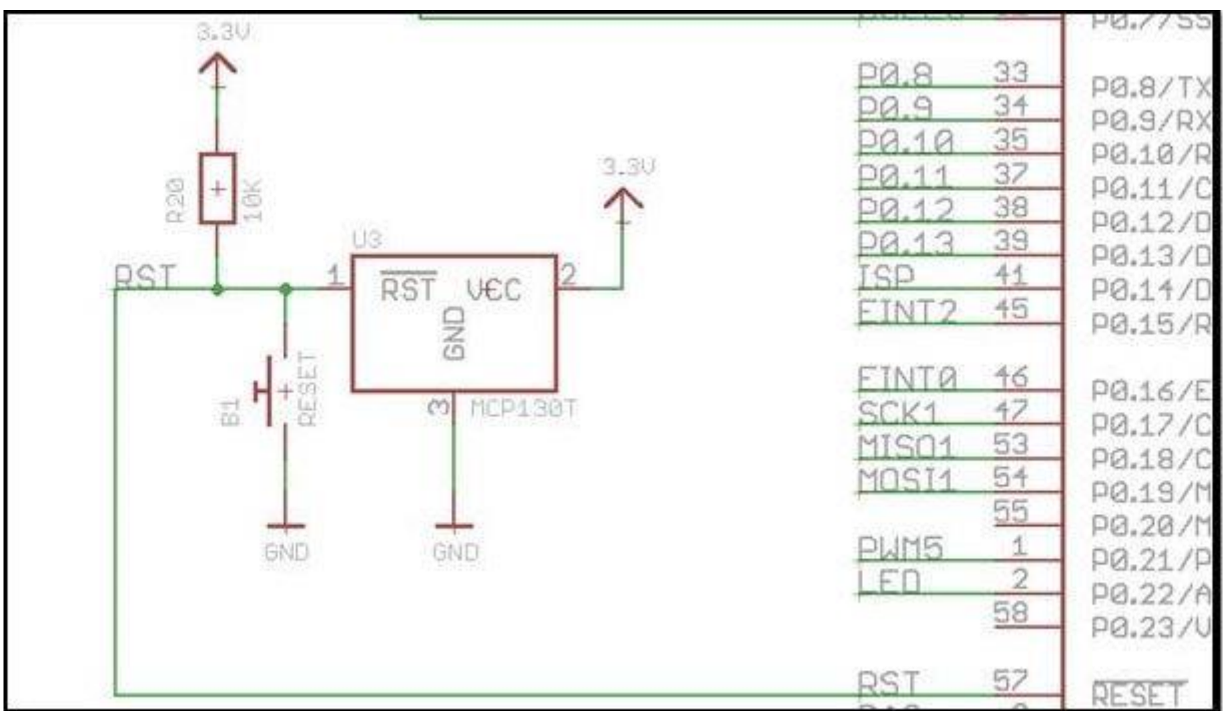
Regulator:

It is used to regulate DC supply output.

2. Reset Circuit

Reset button is essential in a system to avoid programming pitfalls and sometimes to manually bring back the system to the initialization mode. Circuit diagram for reset is as shown below.

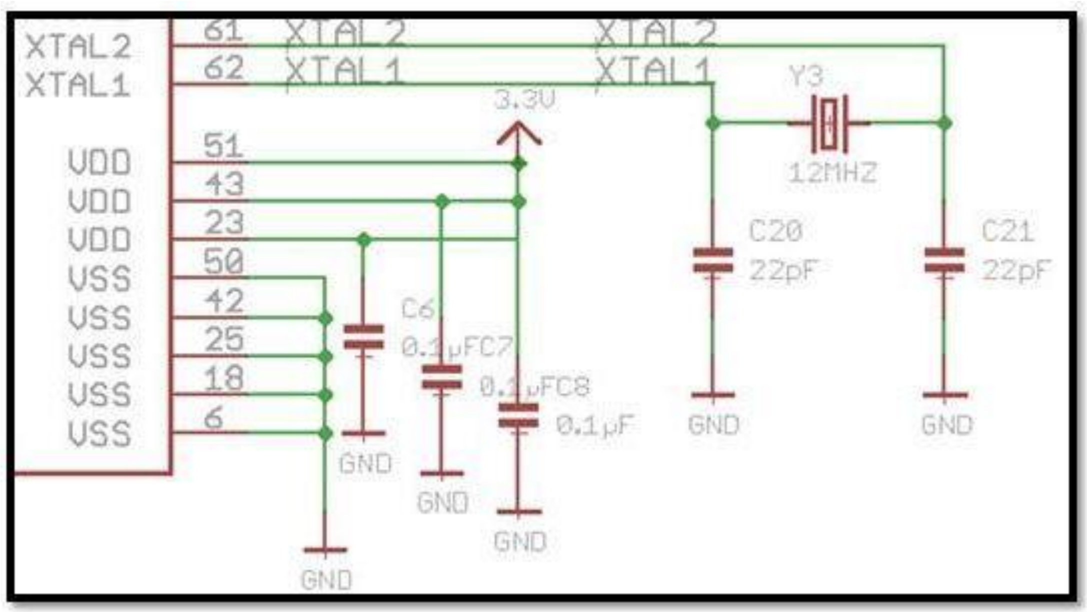
MCP 130T is a special IC used for providing stable RESET signal to LPC 2148.



3.4 RESET CIRCUIT

3. Oscillator Circuit

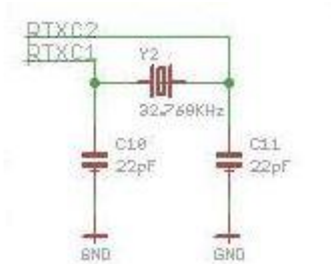
Oscillations, the heartbeat, are provided using a crystal and are necessary for the system to work.



3.5 OSCILLATOR CIRCUIT

4. RTC Oscillator Circuit

It provides clock for RTC operation.

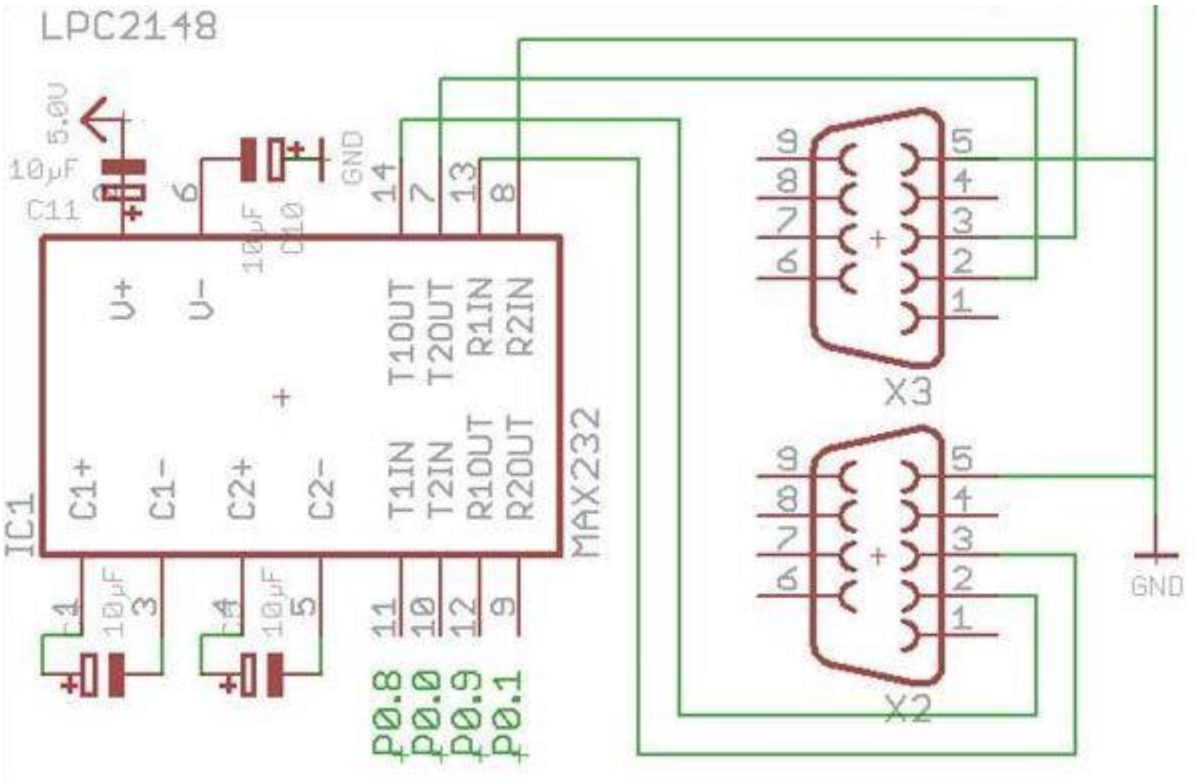


3.6 RTC OSCILLATOR CIRCUIT

5. UART

LPC 2148 has inbuilt ISP which means we can program it within the system using serial communication on COM0. It has also COM1 for serial communication.

MAX 232/233 IC must be used for voltage logic conversion. Related connections are as given below.



3.7 UART

II.ADXL 345

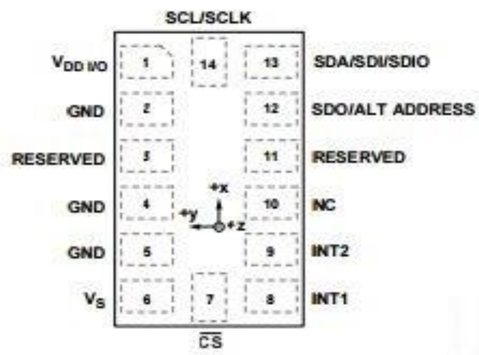
The ADXL345 is a small, thin, low power, 3-axis MEMS accelerometer with high resolution (13-bit) measurement at up to +-16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface.

The ADXL345 is well suited to measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0 degrees.

FEATURES

- 2.0-3.6VDC Supply Voltage
- Ultra Low Power: 40uA in measurement mode, 0.1uA in standby@ 2.5V
- Tap/Double Tap Detection
- Free-Fall Detection

PIN CONFIGURATION

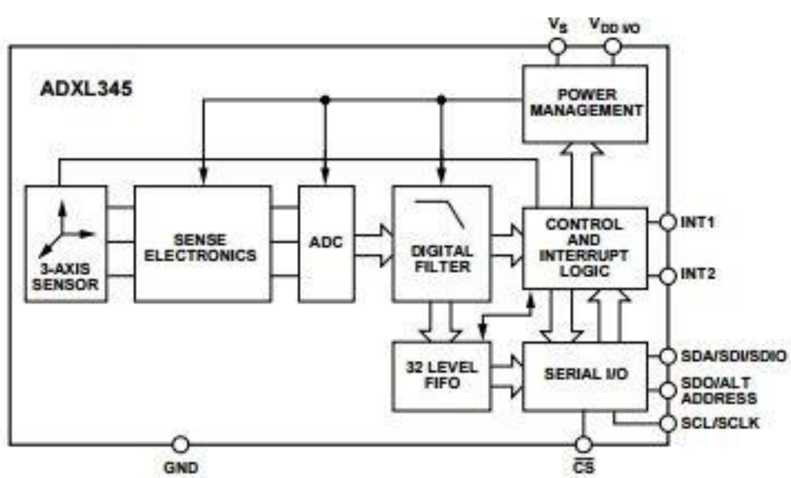


3.8 PIN CONFIGURATION

3.1 PIN FUNCTIONS

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	Must be connected to ground.
3	Reserved	Reserved. This pin must be connected to V_S or left open.
4	GND	Must be connected to ground.
5	GND	Must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	Reserved	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output/Alternate I^2C Address Select.
13	SDA/SDI/SDIO	Serial Data (I^2C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock.

FUNCTIONAL BLOCK DIAGRAM



3.9 FUNCTIONAL BLOCK DIAGRAM

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion and if the acceleration on any axis exceeds a user-set level. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to one of two interrupt output pins. An integrated, patent pending 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor intervention. Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

APPLICATIONS

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection
- Fitness equipment

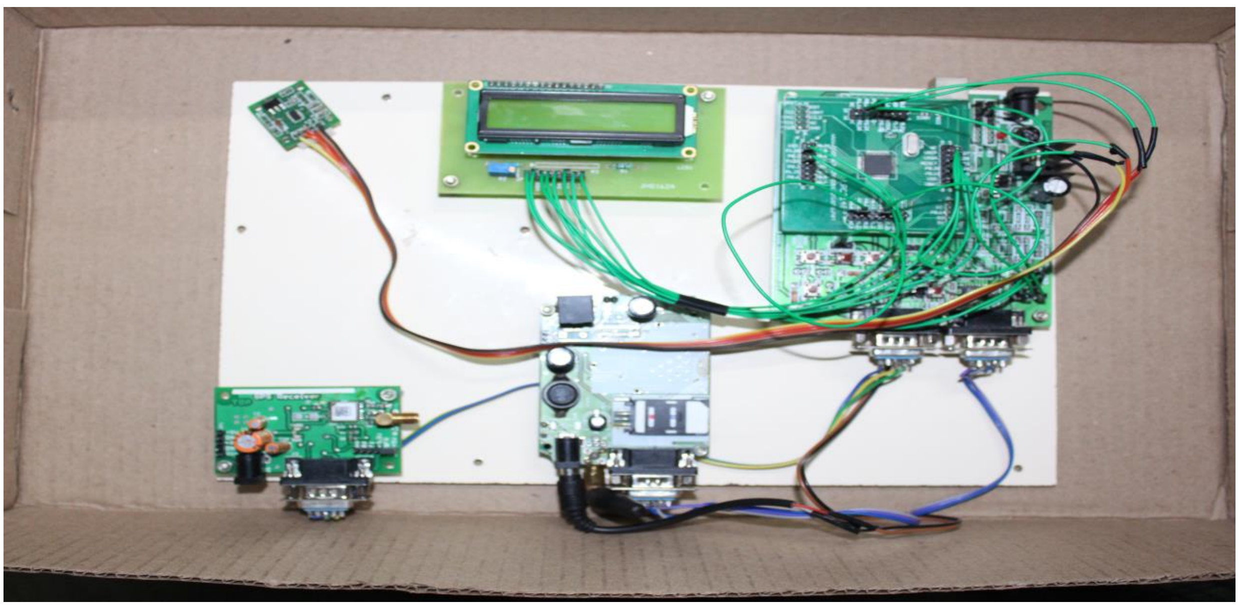
III. GPS MODULE

The Global Positioning System (GPS) is a satellite based navigation system that sends and receives radio signals. A GPS receiver acquires these signals and provides the user with information. Using GPS technology, one can determine location, velocity and time, 24 hours a day, in any weather conditions anywhere in the world for free. GPS was formally known as the NAVSTAR (Navigation Satellite Timing and Ranging). Global Positioning System was originally developed for military. Because of its popular navigation capabilities and because GPS technology can be accessed using small, inexpensive equipment, the government made the system available for civilian use. The USA owns GPS technology and the Department of Defense maintains it. The basis of the GPS technology is a set of 24 satellites that are continuously orbiting the earth. These satellites are equipped with atomic clocks and send out radio signals as to the exact time and their location. These radio signals from the satellites are picked up by the GPS receiver. Once the GPS receiver locks on to four or more of these satellites, it can triangulate its location from the known positions of the satellites. It is a high performance, low power satellite based model. It is a cost effective and portable system which accurately detects the location .

IV. GSM MODULE

GSM (Global System for Mobile communications) is an open, digital cellular technology used for transmitting mobile voice and data services. GSM (Global System for Mobile communication) is a digital mobile telephone system that is widely used in Europe and other parts of the world. GSM uses a variation of Time Division Multiple Access (TDMA) and is the most widely used of the three digital wireless telephone technologies (TDMA, GSM, and CDMA. It operates at either the 900 MHz or 1,800 MHz frequency band. It supports voice calls and data transfer speeds of up to 9.6 kbit/s, together with the transmission of SMS (Short Message Service). The message sending module is SIM900, it is a Tri-band GSM/GPRS that works on frequencies EGSM 900 MHz, DCS 1800 MHz and PCS1900 MHz. SIM900 provides GPRS multi-slot class 10/ class 8 (optional) capability and supports the GPRS coding schemes. The SIM900 provides RF antenna interface with two alternatives: antenna connector and antenna pad. The antenna connector is MM9329-2700. And customer's antenna can be soldered to the antenna pad. The SIM900 is designed with power saving technique, the current consumption to as low as 2.5mA in SLEEP mode. The SIM900 is integrated with the TCP/IP protocol, Extended TCP/IP AT commands are developed for customers to use the TCP/IP protocol easily, which is very useful for data transfer applications. Both GPS and GSM are interfaced to the control unit using serial communication protocol.

3.2 FIRST DESIGN



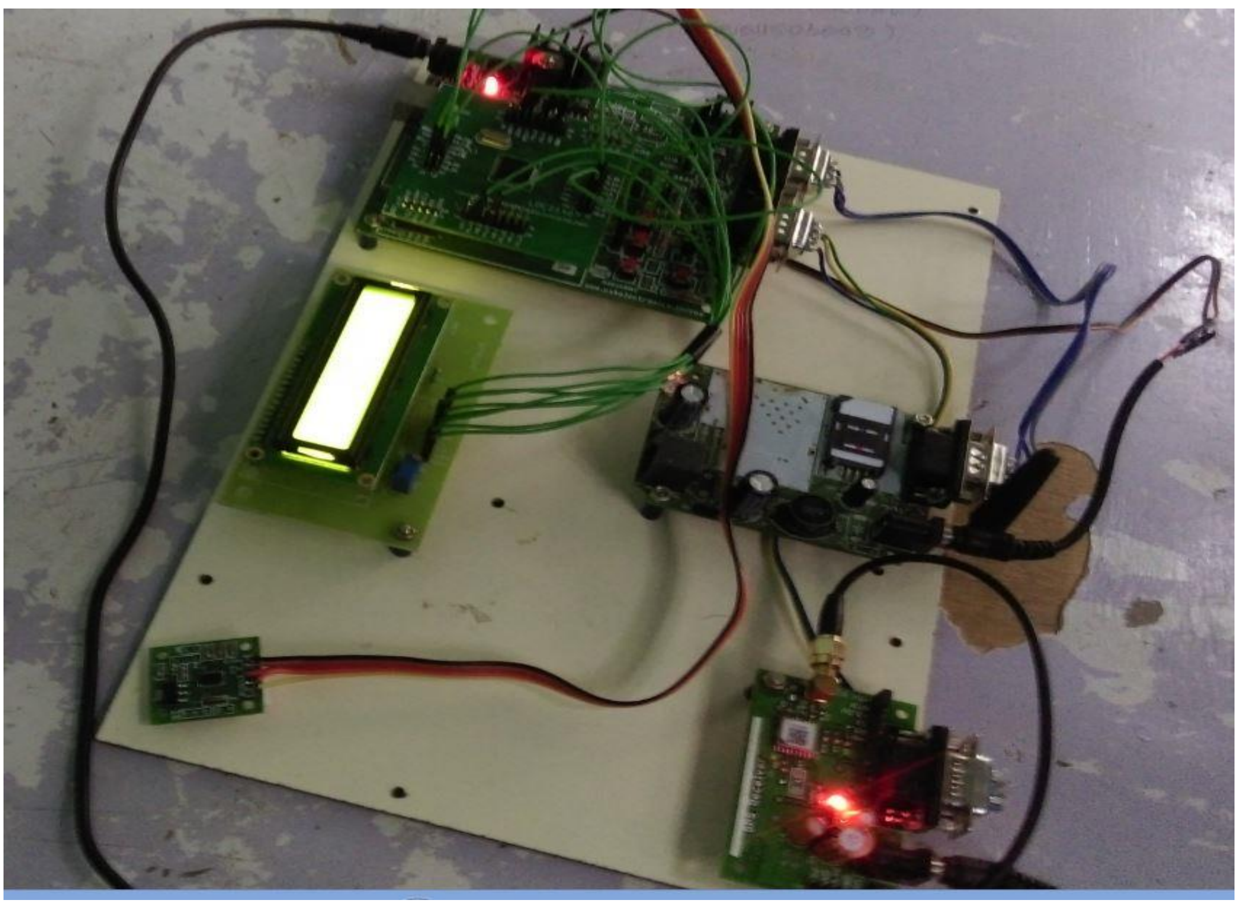
3.10 FIRST DESIGN

CHAPTER 4

RESULTS AND DISCUSSION

4.1 HARDWARE IMPLEMENTATION

At first , hardware connections are given . The antenna connections are given to GPS module. The interface connections between ARM and GPS / GSM module is given by RS 232 . Two adapter module are given to ARM and GPS module. The power supply for GSM SIM 900 can be obtained from ARM processor itself. The ARM processor is made idle while dumping the code by making the pin connected to ground. The LCD used is a 4 bit LCD display and the connections are given accordingly. The voltage pins and ground pins are interfaced accordingly . Various pins from PORT 0 and PORT 1 in the ARM Processor are assigned accordingly. Then , ADXL pins are assigned and connected accordingly, Overall connection of the module is below as follows .



4.1 OVERALL CONNECTION

4.2 PERFORMANCE OF THE PROJECT

I) NORMAL CONDITION

When ADXL 345 is not displaced , the condition will be no fall and the arm processor processes until it gets input from ADXL sensor . Until then GPS retains the Latitude and Longitude values . This is the normal condition.

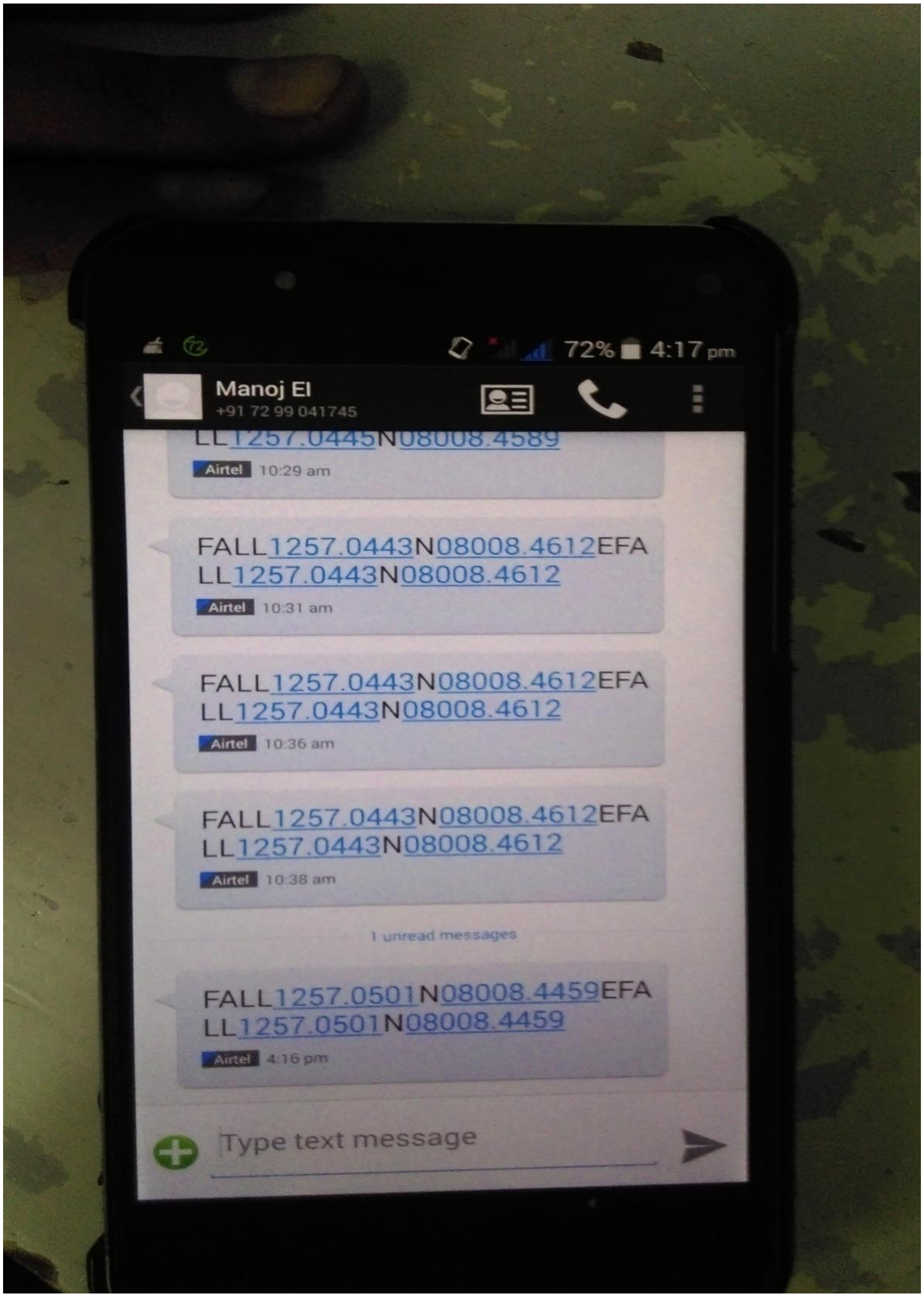
4.2 NOT FALL CONDITION

II) DISPLACED POSITION FOR MAJOR ACCIDENT

For a major accident , ADXL 345 will be displaced and the information is given to ARM Processor . ARM processor will send signal to the GSM Module to send short message service. The Latitude and Longitude are indicated in the LCD Display.

4.3 FALL CONDITION

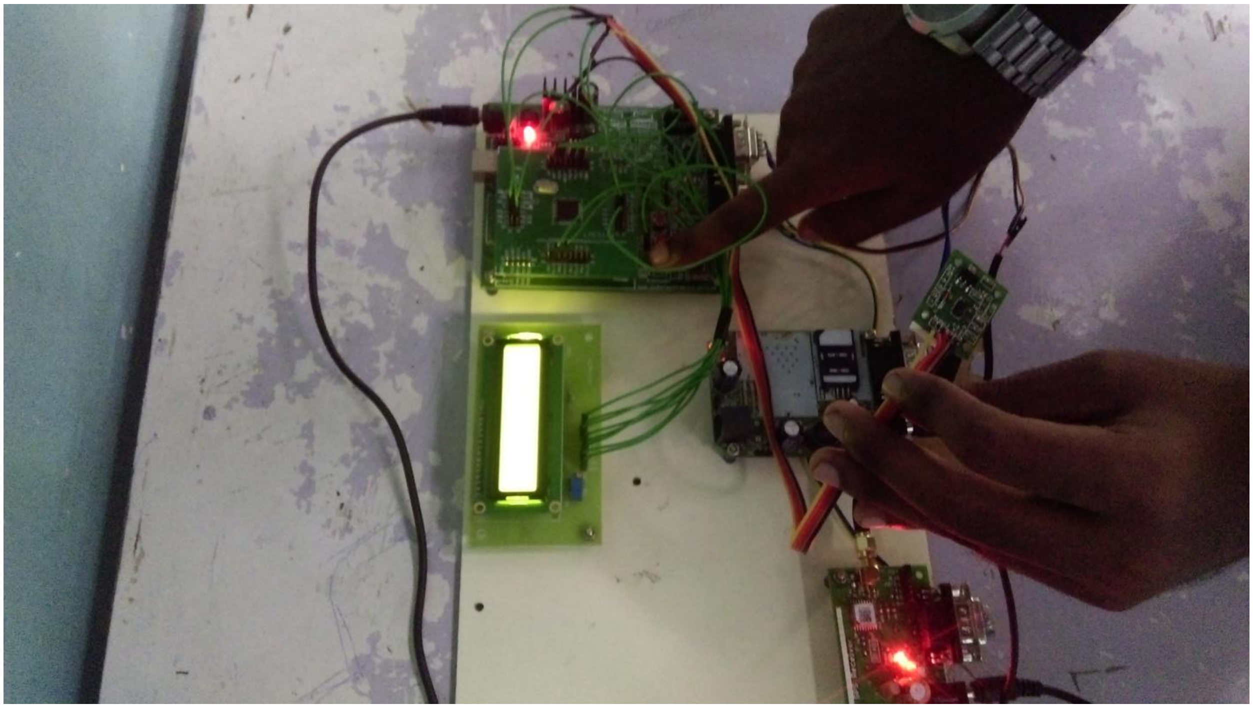
The message is generated with the help of SIM module . The Message is sent to the numbers which are specified in the code along with co-ordinates details.



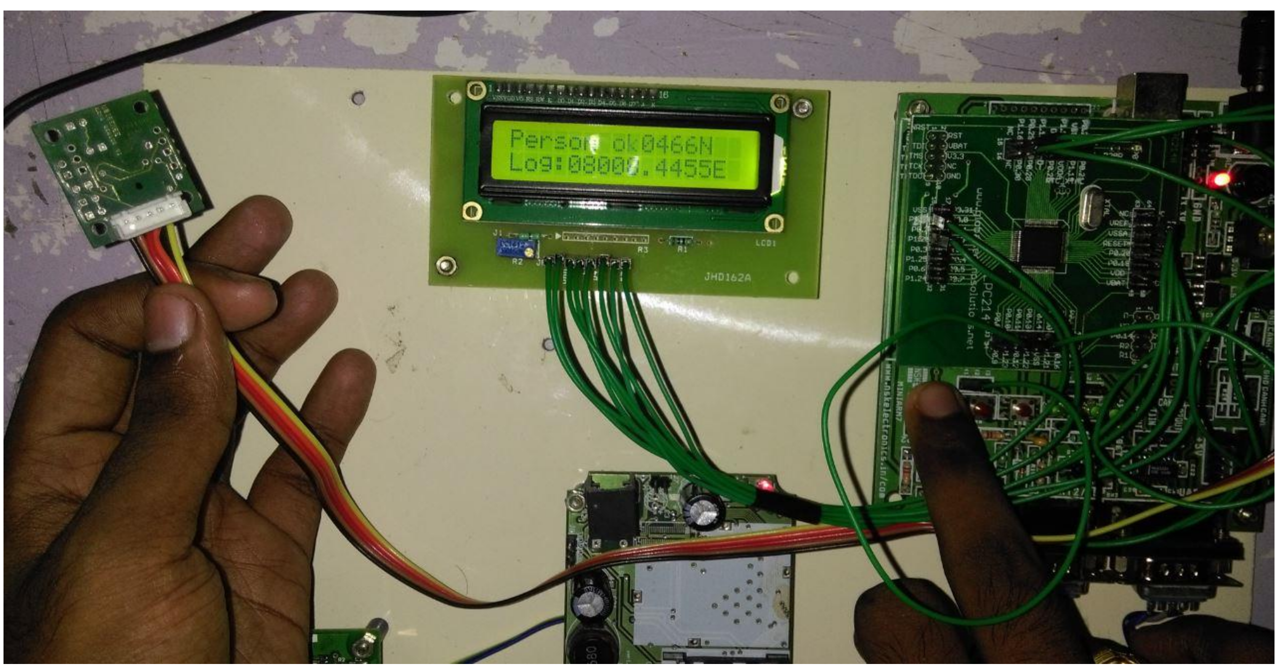
4.4 TEXT MESSAGE WITH CO ORDINATES

III) CONDITION FOR MINOR ACCIDENT

For minor Accident , the message should be interrupted by pressing the switch. The screen will display as person is okay .



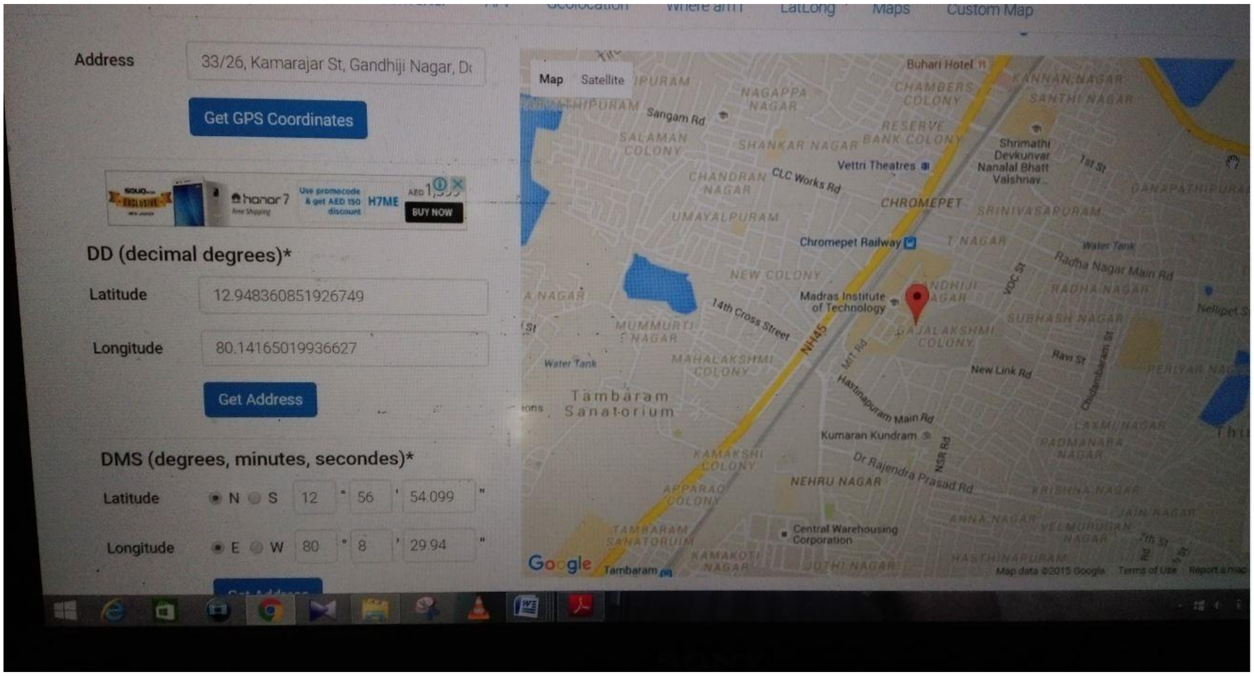
4.5 USAGE OF SWITCH



4.6 PERSON OKAY CONDITION

IV) VERIFICATION

The co ordinates can be checked in the Google to find the exact location.



4.7 CO-ORDINATES VERIFICATION

V) ANTENNAE POSITION

The antennae position must be outside for better speed and better results. It can be kept in an open environment.

4.8 ANTENNAE POSITION

CHAPTER 5

CONCLUSION

5.1 SUMMARY

With the advent of science and technology in every walk of life the importance of vehicle safety has increased and the main priority is being given to reduce the alarming time when an accident occur, so that the wounded lives can be attended in lesser time by the rescue team. This project provides the design which has the advantages of low cost, portability, small size and easy expansibility. The platform of the system is ARM along with Vibration sensor, GPS and GSM, interfacing which shortens the alarm time to a large extent and locate the site of accident accurately. This system can overcome the problems of lack of automated system for accident location detection. Consequently, the time for searching the location is reduced and the person can be treated as soon as possible which will save many lives.

This system will have broad application prospects as it integrates the positioning systems and the network of medical based services. The accident can be detected by vibration sensor which will give the accurate information. The controller will process the data, as soon as input is received by the controller the message is sent through the GSM module. The geographical coordinates and the time of the site of the accident is detected by the GPS module.

An alternate condition is given by pressing a switch, in order to interrupt the flow of sending the message in case of no casualty; this will help to save time of medical rescue team and unnecessary alarming which creates havoc in such unusual conditions. The accident location automatic detection will help us to provide security to the vehicles and to the lives of the people. The high priority is given to the lives of the people. Hence, this paper provides a feasible solution to traffic hazards and it gives security to vehicle and reduces loss of valuable lives and property.

5.2 FUTURE SCOPE

For further inputs , the roll over sensor or MEMS sensor can be deployed in the vehicle . It will provide the further inputs . The future idea can be like, the message can be sent to nearby hospital where the accident has happened , by identifying the co ordinates .

REFERENCES

- International Journal of Reconfigurable and Embedded Systems (IJRES) volume 1 No. 2 , July 2012 , on "Vehicle Accident Automatic detection and remote alarm device "
- Yao Jin. The discussion of Road Traffic Safety Countermeasures System[J]. Private Science and Technology, 2010.
- Wang Wei. Embedded Microcontroller MC9S08AW60 Principles and Application [M]. Beijing: Beijing Aerospace University Press, 2008.
- Zhu Yi, Yang Shubo. MMA series of acceleration sensor principles and application [J]. Medical equipment, 2008.
- TAYLORRK, SCHROCKMD, BLOOMFIELDJ, Dynamic testing of GPS receivers [J]. Transactions of the ASAE,2004,47 (4).
- SIEMENS, TC35i Hardware Interface Description [M]. April 14,2003
- Ma Chao. Embedded GSM message interface hardware and software design [J]. Microcontroller and EmbeddedSystems, 2003,
- Wang Wei and Fan Hanbo, Traffic accident automatic detection and remote alarm device, IEEE proc.ICEICE pp-910-913, April 2011.
- Raj Kamal, EMBEDDED SYSTEMS architecture, programming and design, Second Edition, TATA McGraw-Hill publishing Company Limited, 2008.
- Steve Furber, ARM SYSTEM-ON-CHIP ARCHITECTURE, SECOND EDITION, Pearson Education Limited,2000.
- GSM User Manual, SIMCOM LTD, August 2006.
- Arm User Manual, Phillips, August 2005.
- GPS Beginner`s Guide, Garmin LTD, 2008.

APPENDIX

ARM CODE

```
#include <LPC214X.H>

 #include<string.h>
 #include<stdio.h>
 #include<math.h>
 #define CLEARSCREEN Lcd_Cmd(0x01)
 //#define Z     (1<<21)
 unsigned char X[5];
 unsigned char Z[5];
 unsigned char Y[5];
int A,B,C;
unsigned char buf[25];
void red();
 void delay_s(unsigned int val);
          unsigned char *p,i,j=2,k[6],g[2];
          void delete(void);
   void Lcd_Cmd(int command);
 void delay1(unsigned int delay);
  void lcd_data(int data);
  void LCD( char *dat, char loc);
   int vv=1;
  char   SIM300_AT[]="AT";
  char   SIM300_ATE0[]="ATE0";

  char   SIM300_IPR[]="AT+IPR=9600";
  char   SIM300_CMGF[]="AT+CMGF=1";
  char   SIM300_CMGS[]="AT+CMGS=";
  char   SIM300_CMGR[]="AT+CMGR=1";
  char   SIM300_CMGD[]="AT+CMGD=1";
  char   SIM300_ATOK[]="OK";
  char   SIM300_MSGASK[]=">";
  char   SIM300_MSGOK[]="+CMGR: ";
  char   SIM300_SENDOK[]="+CMGS:";
  char   SIM300_READ[]="REC READ";
```

```
 char   SIM300_UNREAD[]="REC UNREAD";
 char   SIM300_ERROR[]="ERROR";
 char   a[]="1ON";
  char  b[]="1OF";
 char   c[]="2ON";
 char   d[]="2OF";
  char   OVER[1];
  char   SINGLE_QUETES[1];
  char   SEND_MSG[1];
  char   HUICHE[1];
void ReadMsg();
void ADC_Init(void);
unsigned char setbit=0;
unsigned char REC;
unsigned char RX_Len;
char TX_Buf[60];
char RX_Buf[45];

 char data_len              = 0;
 int TX_Len  = 0;
 int TX_Max_Len             = 0;
 unsigned char Close;
 char  TX_Enable;
 void Lcd_Init(void);
void lcdcmd(unsigned char command);
//  void lcd_data(unsigned char);
void lcdstring_disp(unsigned char* string);
char RX_Buf[45];
char control_data[16];
char Out[15];
char carr                  = 0;
char  TX_Enable;
char RX_Enable;
char Ok;
char Send_Enable;
char check;
char count;
char carr2;
char data_send;
char data_send1;
```

```
 char message;
 char web_error;
 char GPRS_Enable;
 int ECG_Sample;
 char data_mode;
 char data_mode1=0;
 unsigned char medicine=0;
/*************** gps**********************/
 char Lat[12]                   = {"0000.0000"};
 char Log[12]                   = {"00000.0000"};
 char RX_Buf_GPS[200];
 char Temp_GPS[100];
 char Buf[4];
 volatile static  unsigned char GPS_Data;
 unsigned char RX_Enable_GPS;
 unsigned char Msg_End;
 unsigned char GGA_Enable;
 unsigned char Close;
 unsigned char RMC;

// void Lcd_Cmd(unsigned char Cmd);
 void Lcd_Delay(unsigned int del);
 char ISR;
 char Conversion_complete;
 char adc_value;
 char LCD_Buf[16];

void Delay(unsigned long delay);
 void UART0_ISR_GPS(void)__irq
 {
   char aa=0;
        aa=U0IIR;
        if(U0LSR==0x60)
         {
                if(GPS_Data == 0x52)
                        {
                                     RX_Len_GPS = 0;
                                          RMC=1;
                        }
                        else if(RMC&&(GPS_Data !=0X0D)&&(GPS_Data != 0X0A))
```

```
                        {
                           RX_Buf_GPS[RX_Len_GPS] = GPS_Data;
                                RX_Len_GPS++;
                        }
                   else  if(GPS_Data==0x0D||GPS_Data==0x0A)
                         {
                          Msg_End = 1;
                          RX_Buf_GPS[RX_Len_GPS]='\0';
                         }
        }
                else if(U0LSR==0x20 )
          {
                                if(TX_Buf[TX_Len]!='\0' && TX_Enable==1)
                                                U0THR=TX_Buf[TX_Len++];
                                else
                                                {
                                                                TX_Len=0;
                                                                TX_Enable=0;
                                                }
          }

        VICVectAddr = 0x00000000;
 }

void UART1(void) __irq
{
  char aa=0;
   if(U1LSR==0x61)
    {
    Close= U1RBR;
         if(Close != 0x0D && Close != 0x0A)
     {
       RX_Buf[RX_Len] = Close;
       RX_Len=RX_Len+1;
     }
    if(Close == 0x0D)
     {
      setbit = 1;
      RX_Len = 0;
      REC++;
```

```
      }
        }
   else if(U1LSR==0x20 )
     {
       if(TX_Buf[TX_Len]!='\0' && TX_Enable==1)
       U1THR=TX_Buf[TX_Len++];
       else
        {
         TX_Len=0;
         TX_Enable=0;
         }
      }
           VICVectAddr = 0x00000000;
  }

void delay(void)
{
  unsigned long wait;
  for(wait=0;wait<=250000;wait++);
}

void Delay(unsigned long delay)
{
 while(delay--);
}

void UART1_Init(void)
{
  U1FCR   = 0x07;
  U1LCR   = 0x83;
  U1DLL   = 0xc3;
  U1DLM   = 0x00;
  U1LCR   = 0x03;
  U1IER   = 0x03;      // tX intr enable
  PINSEL0|= 0x50000;
  VICIntSelect   = 0x00000000;
  VICIntEnable   = 0x00000080;
  VICVectCntl2    = 0x00000020|7;
  VICVectAddr2   = (unsigned int) UART1;
}
```

```
void UART0_Init(void)
{
   PINSEL0|= 5;
//    APBDIV         = 2;
        U0FCR        = 0x07;
        U0LCR        = 0x83;

        U0DLL        = 0xc3;
        U0DLM        = 0x00;
        U0LCR        = 0x03;
        U0IER        = 0x03;          // tX intr enable
   VICIntSelect      = 0x00000000;
        VICIntEnable        = 0x00000040;
        VICVectCntl3        = 0x00000020|6;
        VICVectAddr3        = (unsigned int) UART0_ISR_GPS;
}
void Start_TX(void)
{
   TX_Enable = 1;
   TX_Len      = 1;
   TX_Max_Len=strlen(TX_Buf);
   U1THR      = TX_Buf[0];
   while(TX_Enable);
}

/*****************************************************************
*********************/
void delay_s(unsigned int val){

  T0MR3  = val*29498500;
  T0EMR &= 0xFFFFFFF7;
  T0TC   = 0x00000000;
  T0TCR  =1;
  while((T0EMR & 0x00000008)==0);
}
void ClrRsBuf(char *p,char j)
{
  unsigned char i;

  for(i=0;i<j;i++,p++)
```

```
  {
    *p=' ';
  }
}

/*void Delay(unsigned long delay)
{
 while(delay--);
}*/

void Send_AT(void)
{
char *p;
while(1)
{
        sprintf(TX_Buf,"%s",SIM300_AT);
        strcat(TX_Buf,"\r");
   Start_TX();
        if(setbit==1)
        {
   setbit = 0;
   p=strstr(RX_Buf,SIM300_ATOK);
        strcpy(LCD_Buf,&RX_Buf[0]);
CLEARSCREEN;
        LCD(LCD_Buf,0XC0);
//      lcdcmd(0xc0);

        Delay(10000);
     break;
        }
        else
        {

        Delay(1000000);
        Delay(1000000);
        Delay(1000000);
        Delay(100000);
        Delay(1000000);
        Delay(100000);
}
```

```
}
}

/******************************************************************
************************/
void Send_ATE0(void)
{
char *p;
while(1)
{
        RX_Len=0;
        sprintf(TX_Buf,"%s",SIM300_ATE0);
        strcat(TX_Buf,"\r");

   Start_TX();
        if(setbit==1)
        {
   setbit = 0;
   p=strstr(RX_Buf,SIM300_ATOK);
        //print_line(&RX_Buf[0],0xc0);
        strcpy(LCD_Buf,&RX_Buf[0]);
        Lcd_Cmd(0x01);
LCD(LCD_Buf,0xc0);
        //lcdcmd(0xc0);
        Delay(10000);
  break;
        }
        else
        {
        Delay(1000000);
        Delay(1000000);
        Delay(1000000);
        Delay(100000);
        Delay(1000000);
   Delay(100000);
}
}
}
```

```
/*****************************************************************
************************/
void SetMode(void)
{
  char *p;
   while(1)
   {
            sprintf(TX_Buf,"%s",SIM300_CMGF);
            strcat(TX_Buf,"\r");
      Start_TX();
                    if(setbit==1)
          {
    setbit = 0;
    p=strstr(RX_Buf,SIM300_ATOK);
          strcpy(LCD_Buf,&RX_Buf[0]);
          Lcd_Cmd(0x01);
LCD(LCD_Buf,0XC0);
//        lcdcmd(0xc0);
          Delay(10000);
      break;
          }
          else
          {
          Delay(1000000);
          Delay(1000000);
          Delay(1000000);
          Delay(100000);
          Delay(1000000);
          Delay(100000);
}
}
}
/*****************************************************************
************************/
void SIM300_init(void)
{
          Lcd_Cmd(0x01);
  LCD("SIM300 init begin!",0x80);
//        lcdcmd(0x80);
   Delay(100);
```

```
  // CLEARSCREEN;
        Lcd_Cmd(0x01);
   Delay(100);
   LCD("AT",0x80);
        //lcdcmd(0x80);
   Send_AT();
    LCD("OK",0x80);
//      lcdcmd(0x80);
   Delay(1000);
   //CLEARSCREEN;
        Lcd_Cmd(0x01);
   Delay(1000);
   LCD("ATE0",0x80);
//      lcdcmd(0x80);
   Send_ATE0();
    LCD("OK",0x80);
//      lcdcmd(0x80);
   Delay(1000);
   //CLEARSCREEN;
        Lcd_Cmd(0x01);
   Delay(1000);

  LCD("AT+CMGF=1",0x80);
 // lcdcmd(0x80);
   SetMode();
  LCD("OK",0x80);
  // lcdcmd(0x80);
   Delay(1000);
        Lcd_Cmd(0x01);
   //CLEARSCREEN;
   Delay(1000);
}
/*****************************************************************
*************************/
void SendMsg(  unsigned char  *MSG)
{
    char *p;
          unsigned char i=20;
              RX_Len=0;
          TX_Buf[0] ='A';
```

```
 TX_Buf[1] ='T';
 TX_Buf[2] ='+';
 TX_Buf[3] ='C';
 TX_Buf[4] ='M';
 TX_Buf[5] ='G';
 TX_Buf[6] ='S';
 TX_Buf[7] ='=';
 TX_Buf[8] ='"';
 TX_Buf[9] ='+';
 TX_Buf[10]='9';
 TX_Buf[11]='1';
 TX_Buf[12]='9';
 TX_Buf[13]='8';
 TX_Buf[14]='4';
 TX_Buf[15]='3';
 TX_Buf[16]='8';
 TX_Buf[17]='4';
 TX_Buf[18]='3';
 TX_Buf[19]='9';
 TX_Buf[20]='1';
 TX_Buf[21]='9';
 TX_Buf[22]='"';
 TX_Buf[23]='\r';
// TX_Buf[24]='\r';
 Start_TX();

Delay(1000000);
 Delay(100000);
 Delay(1000000);
 Delay(100000);
 Delay(100000);
Delay(100000);
 while(i--)
 {
                if(setbit==1)
 {
setbit = 0;
p=strstr(RX_Buf,SIM300_MSGASK);
 //print_line(&RX_Buf[0],0xc0);
 strcpy(LCD_Buf,&RX_Buf[0]);
```

```
        Lcd_Cmd(0x01);
LCD(LCD_Buf,0xc0);
//      lcdcmd(0xc0);
        Delay(10000);
        Lcd_Cmd(0x01);
        }
      p=strstr(RX_Buf,SIM300_MSGASK);
      if(p != NULL)
      {
              Delay(100);
          sprintf(TX_Buf,"%s",MSG);
                  strcat(TX_Buf,SEND_MSG);
                   Start_TX();
        Delay(1000000);
        Delay(100000);
        Delay(1000000);
        Delay(100000);
        Delay(100000);
         Delay(100000);
      }
   }
}
void SendMsg1(  unsigned char  *MSG)
{
    char *p;
         unsigned char i=20;
         RX_Len=0;
         TX_Buf[0] ='A';
         TX_Buf[1] ='T';
         TX_Buf[2] ='+';
         TX_Buf[3] ='C';
         TX_Buf[4] ='M';
         TX_Buf[5] ='G';
         TX_Buf[6] ='S';
         TX_Buf[7] ='=';
         TX_Buf[8] ='"';
         TX_Buf[9] ='+';
         TX_Buf[10]='9';
         TX_Buf[11]='1';
         TX_Buf[12]='8';
```

```
          TX_Buf[13]='8';
          TX_Buf[14]='7';
          TX_Buf[15]='0';
          TX_Buf[16]='8';
          TX_Buf[17]='5';
          TX_Buf[18]='6';
          TX_Buf[19]='5';
          TX_Buf[20]='4';
          TX_Buf[21]='1';
          TX_Buf[22]='"';
          TX_Buf[23]='\r';
         // TX_Buf[24]='\r';
          Start_TX();
          Delay(1000000);
         Delay(100000);
         Delay(1000000);
         Delay(100000);
         Delay(100000);
   Delay(100000);
    while(i--)
    {
                                if(setbit==1)
         {
   setbit = 0;
   p=strstr(RX_Buf,SIM300_MSGASK);
         //print_line(&RX_Buf[0],0xc0);
          strcpy(LCD_Buf,&RX_Buf[0]);
         Lcd_Cmd(0x01);
LCD(LCD_Buf,0xc0);
         //lcdcmd(0xc0);
         Delay(10000);
Lcd_Cmd(0x01);
         }
      p=strstr(RX_Buf,SIM300_MSGASK);
      if(p != NULL)
      {
         Delay(100);
           sprintf(TX_Buf,"%s",MSG);
                    strcat(TX_Buf,SEND_MSG);
                     Start_TX();
```

```
        Delay(1000000);
        Delay(100000);
        Delay(1000000);
        Delay(100000);
        Delay(100000);
        Delay(100000);
      }
   }
}
void SendMsg2(  unsigned char  *MSG)
{
    char *p;
         unsigned char i=20;
         TX_Buf[0] ='A';
         TX_Buf[1] ='T';
         TX_Buf[2] ='+';
         TX_Buf[3] ='C';
         TX_Buf[4] ='M';
         TX_Buf[5] ='G';
         TX_Buf[6] ='S';
         TX_Buf[7] ='=';
         TX_Buf[8] ='"';
         TX_Buf[9] ='+';
         TX_Buf[10]='9';
         TX_Buf[11]='1';
         TX_Buf[12]='8';
         TX_Buf[13]='6';
         TX_Buf[14]='7';
         TX_Buf[15]='5';
         TX_Buf[16]='3';
         TX_Buf[17]='5';
         TX_Buf[18]='6';
         TX_Buf[19]='5';
         TX_Buf[20]='8';
         TX_Buf[21]='0';
         TX_Buf[22]='"';
         TX_Buf[23]='\r';
        // TX_Buf[24]='\r';
         Start_TX();
```

```
   Delay(1000000);
        Delay(100000);
        Delay(1000000);
        Delay(100000);
        Delay(100000);
   Delay(100000);
    while(i--)
    {
                            if(setbit==1)
        {
   setbit = 0;
   p=strstr(RX_Buf,SIM300_MSGASK);
        //print_line(&RX_Buf[0],0xc0);
         strcpy(LCD_Buf,&RX_Buf[0]);
        Lcd_Cmd(0x01);
        LCD(LCD_Buf,0xc0);
        //lcdcmd(0xc0);
        Delay(10000);
        Lcd_Cmd(0x01);
        }
      p=strstr(RX_Buf,SIM300_MSGASK);
      if(p != NULL)
      {
        Delay(100);
          sprintf(TX_Buf,"%s",MSG);
                  strcat(TX_Buf,SEND_MSG);
                   Start_TX();
          Delay(1000000);
        Delay(100000);
        Delay(1000000);
        Delay(100000);
        Delay(100000);
          Delay(100000);
      }
   }
}

/*****************************************************************
************************/
unsigned char  gps_receiver(void)
```

```
{
    while(1)
          {
                 if(Msg_End)
            {
        Msg_End = 0;
                     RX_Len_GPS=0;
                     strcpy(Temp_GPS,RX_Buf_GPS);
                     if(Temp_GPS[0] == 'M')
                     {
                            unsigned char Comm = 0;
                            unsigned char i = 0;
                            while(Comm <7)
                            {
                                   if(Temp_GPS[i] == 0x2c)
                                   {
                                          Comm += 1;
                                          if(Comm == 2)
                                          {

                                                 Buf[0] = Temp_GPS[++i];
                                    Buf[1] = 0;
                                                 if(Buf[0] == 0X41)
                                                 {
                                                   RMC_Enable = 1;
                                                 }
                                                 else
                                                 {
                                                        RMC_Enable = 0;
                                            LCD("    Tracking  ",0x80);
                                                 //lcdcmd(0xc0);
                                                 LCD("   Satellites  ",0xc0);
                                                 }
                                                 i-= 2;
                                           // lcdcmd(0x8e);
                                          LCD(Buf,0x8e);
                                          }
                                          i++;
                                   }
                                   else
```

```
        i++;
}
Comm = 0;
if(RMC_Enable == 1)
{
        unsigned char Comm=0;
        unsigned char k=0;
        unsigned char k1=0;
        unsigned char i = 0;
        RMC_Enable = 0;
Send_Enable = 1;
        while(Comm < 6)
        {
                if(Temp_GPS[i] == 0x2c)
                {
                        Comm += 1;
                        i++;
                }
                else
                   i++;
                if(Comm == 3)
                {

                        Lat[k++] = Temp_GPS[i];
                }
                else if(Comm == 5)
                {
                        Log[k1++] = Temp_GPS[i];
                }
        }
        Lat[--k] = 'N';
        Lat[++k] = 0;

        Log[--k1] = 'E';
        Log[++k1] = 0;
//      CLEARSCREEN;
//      lcdcmd(0x80);
LCD("                ",0x80);
//      lcdcmd(0x80);
   LCD("Lat:",0x80);
```

```
                    LCD(Lat,0x84);
        // lcdcmd(0xc0);;
                    LCD("            ",0xc0);
                //      lcdcmd(0xc0);
                    LCD("Log:",0xc0);
                    LCD(Log,0xc4);

                        return 1;
                }
            }
        }
    }

}
void delay1(unsigned int delay)
{
unsigned int i,j;
for(i=0;i<delay;i++)
{
for(j=0;j<65535;j++)
{}
}
}

void Lcd_Init(void)
{
 IO1DIR=0XFFFFFFFF;
 Lcd_Cmd(0x02);
 Lcd_Cmd(0x28);
 Lcd_Cmd(0x01);
 Lcd_Cmd(0x06);
 Lcd_Cmd(0x0d);
 Lcd_Cmd(0x0e);
 Lcd_Cmd(0x80);
 Lcd_Cmd(0xc0);
}
/*****************************************************************
*******************/
void Lcd_Cmd(int command)
{
```

```
int Temp_command,Temp1_command,Temp2_command;
IO1PIN =0X10000;
Temp_command=command;
IO1PIN=Temp_command;
Temp1_command=Temp_command<<24;
IO1PIN=Temp1_command;
Temp1_command=Temp1_command&0XF0000000;
IO1PIN=Temp1_command;
Temp1_command=Temp1_command|0X08000000;
IO1PIN=Temp1_command;
delay1(3);
IO1PIN=Temp1_command^0X08000000;
Temp_command=command;
Temp2_command=Temp_command<<28;
Temp2_command=Temp2_command&0XF0000000;
IO1PIN=Temp2_command;
Temp2_command=Temp2_command|0X08000000;
IO1PIN=Temp2_command;
delay1(3);
IO1PIN=Temp2_command^0x08000000;
IO1PIN = 0X00000;
}
/*****************************************************************
*******************/
void lcd_data(int data)
{
 int Temp_data,Temp1_data,Temp2_data;
Temp_data=data;
Temp1_data=Temp_data<<24;
Temp1_data=Temp1_data&0XF0000000;
IO1PIN=Temp1_data;
Temp1_data=Temp1_data|0X0C000000;
IO1PIN=Temp1_data;
delay1(3);
IO1PIN=Temp1_data^0X0C000000;
Temp_data=data;
Temp2_data=Temp_data<<28;
Temp2_data=Temp2_data&0XF0000000;
IO1PIN=Temp2_data;
Temp2_data=Temp2_data|0X0C000000;
```

```
IO1PIN=Temp2_data;
delay1(3);
IO1PIN=Temp2_data^0x0C000000;
}
/*****************************************************************
*******************/
 void LCD( char *dat, char loc)
{
 Lcd_Cmd(loc);
 IO1PIN = 0X20000;
 while(*dat)
 {
   lcd_data(*dat);
   dat++;
  }
   IO1PIN=0X00000;
 }
int main(void)
{
  int A,B,C;

             unsigned int newButtonState;
      int k=0;
           unsigned char  E[]=40;
            unsigned char F[]=70;

            OVER[0] = 0x0d ;
      SINGLE_QUETES[0] = 0x22;
      SEND_MSG[0] = 0x1a;
      HUICHE[0] = 0x0a;
            Lcd_Init();
           ADC_Init();

      UART1_Init();
      UART0_Init();
             IO0DIR=(1<<21)|(1<<17)|(1<<10);
             IO0CLR=(1<<10);
             LCD("GSM Based Vehicle",0x80);
 LCD("Monitoring System",0xC0);
CLEARSCREEN;
```

```
      Delay(5000);
                  SIM300_init();
 while(1)
 {
        gps_receiver();
        //CLEARSCREEN;
   A=adc_read(1);              //P0.28
    sprintf(X,"%d",A);

         B =adc_read(2);           //P0.29
    sprintf(Y,"%d",B);

         C=adc_read(3);                              //P0.30
    sprintf(Z,"%d",C);
                  if(A > C)
{
if(IO0PIN&(1<<10))
           {
             LCD("Person ok",0x80);
                  Delay(1000);
       Lcd_Cmd(0x01);
                  Delay(1000);
        vv=0;
                  }
                  if(vv)
           {
                  Lcd_Cmd(0x01);
       LCD("FALL",0x80);

             SIM300_init();
             strcpy(buf,"FALL");
                  strcat(buf,Lat);
                  strcat(buf,Log);
                  SendMsg(buf);
                  delay();
                  for(k=0;buf[k]!='\0';k++)
                  {
                  buf[k]=0;
                  }
```

```
            Delay(1000);
             SIM300_init();
              strcpy(buf,"FALL");
                    strcat(buf,Lat);
                    strcat(buf,Log);
                    SendMsg1(buf);
                    delay();
                    for(k=0;buf[k]!='\0';k++)
                    {
                    buf[k]=0;
                    }
            Delay(1000);
             SIM300_init();
              strcpy(buf,"FALL");
                    strcat(buf,Lat);
                    strcat(buf,Log);
                    SendMsg2(buf);
                    delay();
                    for(k=0;buf[k]!='\0';k++)
                    {
                    buf[k]=0;
                    }
            Delay(1000);
            LCD("Monitoring...",0x80);
            Delay(1000);
       Lcd_Cmd(0x01);
                   }
          }
          else
          {
          Lcd_Cmd(0x01);
          LCD(" NOT FALL",0x80);
          }
}
}
void ADC_Init(void)
 {
 //PINSEL0 |= ((1<<8)|(1<<9)|(1<<10)|(1<<11));    //AD0.6+AD0.7
  PINSEL1=(1<<18)|(0<<19)|(0<<25)|(0<<27)|(1<<28)|(0<<29);
 VPBDIV  = 2;
```

```
}
 int adc_read(char ch)
{
 int i;
 long regVal;
AD0CR = 0x00200D01 | (1<<ch);
AD0CR |= 0x01000000;                // Start A/D Conversion
if ( regVal & 0x0000FF00 )       /* check OVERRUN error first */
      {
             regVal = AD0GDR;
            }
        do
         {
           i= AD0GDR;                             // Read A/D Data Register
         } while ((i & 0x80000000)!= 0x80000000); // Wait for end of A/D
Conversion


                     AD0CR &= 0xF8FFFFFF;                   // Stop A/D Conversion
               return(i=(i >> 6) & 0x03FF);                        // bit 6:15 is 10 bit AD
value
                     }
```