CHAPTER 1

INTRODUCTION

In chapter 1 we discuss about introduction of Embedded Systems, Problem Statement, Project Objective, Project Scope and Project Outline

1.1 INTRODUCTION

This project aims at protecting and avoiding gas leak. This project consists of ZIGBEE transmitter section and ZIGBEE receiver section. In ZIGBEE transmitter section it consists of sensor unit, display unit and controller unit.

The sensor detects the gas leak and sends the information to the receiver controller unit PC and it display on PC where the gas leak. In ZIGBEE transmitter section it consists of Alarm unit, controller unit, and display unit.

If the sensor in the transmitter section sends the information to the controller and it immediately display where the gas leak and alarm will be on that particular unit .This device will protect the person using the kitchen from indulging a fire accident as well as it do safety measure on gas leak automatically and safely.

1.2 PROBLEM STATEMENT

In the large petrochemical industry, one of the most concerning problem is the leakage of toxic gas. To solve this problem, it is necessary to locate the leak points and feed the possible location of leak points back to rescuers. Although some researchers have previously presented several methods to locate leak points, they ignored the impact of external factors, such as wind, and internal factors, such as the internal pressure of equipment, on the accurate detection of leak points. Thus here we are using the gas sensor which placed in the leak points, which senses the concentration value of toxic gases such as carbon mono-oxide and it imitates to the mobile device when the concentration value exceeds the normal value. The signal given to the ARM7 microcontroller which intimates the mobile device through zigbee communication module. This article proposes context-aware system architecture for leak point detection in the large scale petrochemical industry. This architecture is a new scheme for accurate leak point detection, which is more consistent with practical application in the large scale petrochemical industry.

1.3 PROJECT OBJECTIVE

The aim of the project is Development of Gas Leak Detection and Location System Based on Wireless Technology. All the devices such as LCD, MAX232, Zigbee, UART,Gas Sensor and PC are being interfacing to microcontroller which forms the control unit of the project. The particular objective of the project is to monitor gas leakage parameter. When they exceeds threshold, intimation is given to the nearby control section including readings of parameter and location of the gas leakage.

1.4 PROJECT SCOPE

This project “Development of Gas Leak Detection and Location System Based on Wireless Technology” is majorly used in Hotels, Gas Stations and more.

In future, more detecting systems like gas detection systems can be implemented.

Here additional features includes more secure systems like a call will be gone to telephone number if a gas leakage d fire in a N/W area like LAN and internet used to worldwide

1.5 PROJECT OUTLINE

The project is organized into 6 chapters, namely introduction, Literature Review, Design approach, Result analysis and conclusion.Chapter2 contains the complete details about the Introduction of Embedded Systems and ATMEL Microcontroller. Chapter 3 describes about the design issues, software and hardware requirements for theImplementation of agricultural automation through webpageChapter 4 consists of the result analysis, applications and advantages. Chapter 5 contains coding and chapter 6 contains conclusion and proposed works to enhance the project in the future.

CHAPTER2

LITERATURE REVIEW

In this chapter we discuss about EMBEDDED SYSTEMS and about the LPC2148and AT89S52 microcontrollers**.**

2.1 INTRODUCTION OF EMBEDDED SYSTEM

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

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things. Many people use the term general-purpose computer to make this distinction clear. As shipped, a general-purpose computer is a blank slate; the manufacturer does not know what the customer will do wish it. One customer may use it for a network file server another may use it exclusively for playing games, and a third may use it to write the next great American novel.

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

At the possible risk of confusing you, it is important to point out that a general-purpose computer is itself made up of numerous embedded systems. For example, my computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive, and sound card-each of which is an embedded system. Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over analog telephone line. That's it and all of the other devices can be summarized in a single sentence as well.

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

2.1.1 History and Feature

Given the definition of embedded systems earlier is this chapter; the first such systems could not possibly have appeared before 1971. That was the year Intel introduced the world's first microprocessor. This chip, the 4004, was designed for use in a line of business calculators produced by the Japanese Company Busicom. In 1969, Busicom asked Intel to design a set of custom integrated circuits-one for each of their new calculator models. The 4004 was Intel's response rather than design custom hardware for each calculator, Intel proposed a general-purpose circuit that could be used throughout the entire line of calculators. Intel's idea was that the software would give each calculator its unique set of features.

The microcontroller was an overnight success, and its use increased steadily over the next decade. Early embedded applications included unmanned space probes, computerized traffic lights, and aircraft flight control systems. In the 1980s, embedded systems quietly rode the waves of the microcomputer age and brought microprocessors into every part of our kitchens (bread machines, food processors, and microwave ovens), living rooms (televisions, stereos, and remote controls), and workplaces (fax machines, pagers, laser printers, cash registers, and credit card readers).

It seems inevitable that the number of embedded systems will continue to increase rapidly. Already there are promising new embedded devices that have enormous market potential; light switches and thermostats that can be central computer, intelligent air-bag systems that don't inflate when children or small adults are present, pal-sized electronic organizers and personal digital assistants (PDAs), digital cameras, and dashboard navigation systems. Clearly, individuals who possess the skills and desire to design the next generation of embedded systems will be in demand for quite some time**.**

2.1.2 Real Time Systems

One subclass of embedded is worthy of an introduction at this point. As commonly defined, a real-time system is a computer system that has timing constraints. In other words, a real-time system is partly specified in terms of its ability to make certain calculations or decisions in a timely manner. These important calculations are said to have deadlines for completion. And, for all practical purposes, a missed deadline is just as bad as a wrong answer.

The issue of what if a deadline is missed is a crucial one. For example, if the real-time system is part of an airplane's flight control system, it is possible for the lives of the passengers and crew to be endangered by a single missed deadline. However, if instead the system is involved in satellite communication, the damage could be limited to a single corrupt data packet. The more severe the consequences, the more likely it will be said that the deadline is "hard" and thus, the system is a hard real-time system. Real-time systems at the other end of this discussion are said to have "soft" deadlines.

All of the topics and examples presented in this book are applicable to the designers of real-time system who is more delight in his work. He must guarantee reliable operation of the software and hardware under all the possible conditions and to the degree that human lives depend upon three system's proper execution, engineering calculations and descriptive paperwork.

2.1.3 Application Areas

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication, data communication, telecommunications, transportation, military and so on.

2.1.3.1 Consumer appliances

At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air-conditioner, VCO player, video game consoles, video recorders etc. Today’s high-tech car has about 20 embedded systems for transmission control, engine spark control, air-conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing.

2.1.3.2 Office automation

The office automation products using embedded systems are copying machine, fax machine, key telephone, modem, printer, scanner etc.

2.1.3.3 Industrial automation

Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs. The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

2.1.3.4 Medical electronics

Almost every medical equipment in the hospital is an embedded system. These equipments include diagnostic aids such as ECG, EEG, blood pressure measuring devices, X-ray scanners; equipment used in blood analysis, radiation, colonoscopy, endoscopy etc. Developments in medical electronics have paved way for more accurate diagnosis of diseases.

2.1.3.5 Computer networking

Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols. For example, a router interconnects two networks. The two networks may be running different protocol stacks. The router's function is to obtain the data packets from incoming pores, analyze the packets and send them towards the destination after doing necessary protocol conversion. Most networking equipments, other than the end systems (desktop computers) we use to access the networks, are embedded systems

2.1.3.6 Telecommunications

In the field of telecommunications, the embedded systems can be categorized as subscriber terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Dissemblers (PADs), sate11ite modems etc. IP phone, IP gateway, IP gatekeeper etc. are the latest embedded systems that provide very low-cost voice communication over the Internet.

2.1.3.7 Wireless technologies

Advances in mobile communications are paving way for many interesting applications using embedded systems. The mobile phone is one of the marvels of the last decade of the 20'h century. It is a very powerful embedded system that provides voice communication while we are on the move. The Personal Digital Assistants and the palmtops can now be used to access multimedia services over the Internet. Mobile communication infrastructure such as base station controllers, mobile switching centers are also powerful embedded systems.

2.1.3.8 Insemination

Testing and measurement are the fundamental requirements in all scientific and engineering activities. The measuring equipment we use in laboratories to measure parameters such as weight, temperature, pressure, humidity, voltage, current etc. are all embedded systems. Test equipment such as oscilloscope, spectrum analyzer, logic analyzer, protocol analyzer, radio communication test set etc. are embedded systems built around powerful processors. Thank to miniaturization, the test and measuring equipment are now becoming portable facilitating easy testing and measurement in the field by field-personnel.

2.1.3.9 Security

Security of persons and information has always been a major issue. We need to protect our homes and offices; and also the information we transmit and store. Developing embedded systems for security applications is one of the most lucrative businesses nowadays. Security devices at homes, offices, airports etc. for authentication and verification are embedded systems. Encryption devices are nearly 99 per cent of the processors that are manufactured end up in~ embedded systems. Embedded systems find applications in every industrial segment-consumer electronics, transportation, avionics, biomedical engineering, manufacturing, process control and industrial automation, data communication, telecommunication, defense, security etc., used to encrypt the data/voice being transmitted on communication links such as telephone lines. Biometric systems using fingerprint and face recognition are now being extensively used for user authentication in banking applications as well as for access control in high security buildings.

2.1.3.10 Finance

Financial dealing through cash and checks are now slowly paving way for transactions using smart cards and ATM (Automatic Teller Machine, also expanded as Any Time Money) machines. Smart card, of the size of a credit card, has a small micro-controller and memory; and it interacts with the smart card reader! ATM machine and acts as an electronic wallet. Smart card technology has the capability of ushering in a cashless society. Well, the list goes on. It is no exaggeration to say that eyes wherever you go, you can see, or at least feel, the work of an embedded system!

2.1.4 Overview of Embedded System Architecture

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also called the 'firmware'. The operating

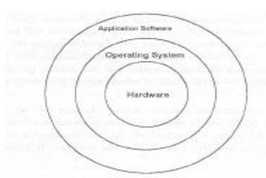


Figure 2.1: Layered architecture of an embedded system

System runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system. For small appliances such as remote control units, air conditioners, toys etc., there is no need foran operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip.Once the software is transferred to the memory chip, the software will continue to run fora long time you don't need to reload new software.

Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig 2.2 the building blocks are:

- Central Processing Unit (CPU)
- Memory (Read-only Memory and Random Access Memory)
- Input Devices
- Output devices
- Communication interfaces
- Application-specific circuitry

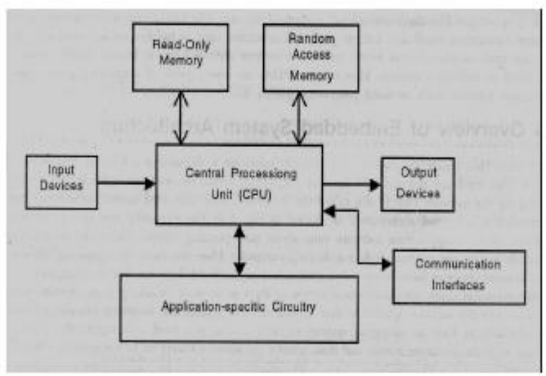


Figure 2.2: A building blocks of an embedded system

2.1.4.1 Central Processing Unit (CPU)

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. D5P is used mainly for applications in which signal processing is involved such as audio and video processing.

2.1.4.2 Memory

The memory is categorized as Random Access 11emory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is program is executed.

2.1.4.3 Input devices

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device foruser interaction; they take inputs fromsensors or transducers 1’fnd produce electrical signals that are in turn fed to other systems.

2.1.4.4 Output devices

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a fewLight Emitting Diodes (LEDs) *to* indicate the health status of the system modules, orforvisual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display someimportant parameters.

2.1.4.5 Communication interfaces

The embedded systems may need to, interact with other embedded systems at they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a *few* communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), and IEEE 1394, Ethernet etc.

2.1.4.6 Application-specific circuitry

Sensors, transducers, special processing and control circuitry may be required fat an embedded system, depending on its application. This circuitry interacts with the processor to carry out the necessary work. The entire hardware has to be given power supply either through the 230 volts main supply or through a battery. The hardware has to design in such a way that the power consumption is minimized.

2.1.4.7 Conclusions

Embedded Systems plays a vital role in our day today life. They are used for household appliances like microwave oven to the satellite applications. They provide good man to machine interface.

Automation is the further step in the world of Embedded Systems, which includes the elimination of the human being in the mundane applications. They are cost effective, accurate and can work in any conditions and round the clock.

2.2 Microcontroller

Microcontrollers as the name suggests are small controllers. They are like single chip computers that are often embedded into other systems to function as processing/controlling unit. For example the remote control you are using probably has microcontrollers inside that do decoding and other controlling functions. They are also used in automobiles, washing machines, microwave ovens, toys ... etc, where automation is needed.

Micro-controllers are useful to the extent that they communicate with other devices, such as sensors, motors, switches, keypads, displays, memory and even other micro-controllers. Many interface methods have been developed over the years to solve the complex problem of balancing circuit design criteria such as features, cost, size, weight, power consumption, reliability, availability, manufacturability. Many microcontroller designs typically mix multiple interfacing methods. In a very simplistic form, a micro-controller system can be viewed as a system that reads from (monitors) inputs, performs processing and writes to (controls) outputs.

Embedded system means the processor is embedded into the required application. An embedded product uses a microprocessor or microcontroller to do one task only. In an embedded system, there is only one application software that is typically burned into ROM. Example: printer, keyboard, video game player

Microprocessor - A single chip that contains the CPU or most of the computer

Microcontroller - A single chip used to control other devices

Microcontroller differs from a microprocessor in many ways. First and the most important is its functionality. In order for a microprocessor to be used, other components such as memory, or components for receiving and sending data must be added to it. In short that means that microprocessor is the very heart of the computer. On the other hand, microcontroller is designed to be all of that in one. No other external components are needed for its application because all necessary peripherals are already built into it. Thus, we save the time and space needed to construct devices.

2.2.1 Advantages of using a Microcontroller over Microprocessor

- Microprocessors are single-chip CPUs used in microcomputers.
- Microcontrollers and microprocessors are different in three main aspects: hardware architecture, applications, and instruction set features.
- Applications: Microprocessors are commonly used as a CPU in computers while microcontrollers are found in small, minimum component designs performing control oriented activities.
- Hardware architecture: A microprocessor is a single chip CPU while a microcontroller is a single IC contains a CPU and much of remaining circuitry of a complete computer (e.g., RAM, ROM, serial interface, parallel interface, timer, and interrupt handling circuit).
- Microprocessor instruction sets are processing Intensive.

- Their instructions operate on nibbles, bytes, words, or even double words.
- Addressing modes provide access to large arrays of data using pointers and offsets.
- They have instructions to set and clear individual bits and perform bit operations.
- They have instructions for input/output operations, event timing, enabling and setting priority levels for interrupts caused by external stimuli.
- Processing power of a microcontroller is much less than a microprocessor.

2.2.2 Applications

Micro controller applications are

- Cell phones.
- Computers.
- Robots.
- Interfacing to two pc.
- Laptops

2.3 ARM 7 MICRO CONTROLLERS

2.3.1 Pin diagram of arm7

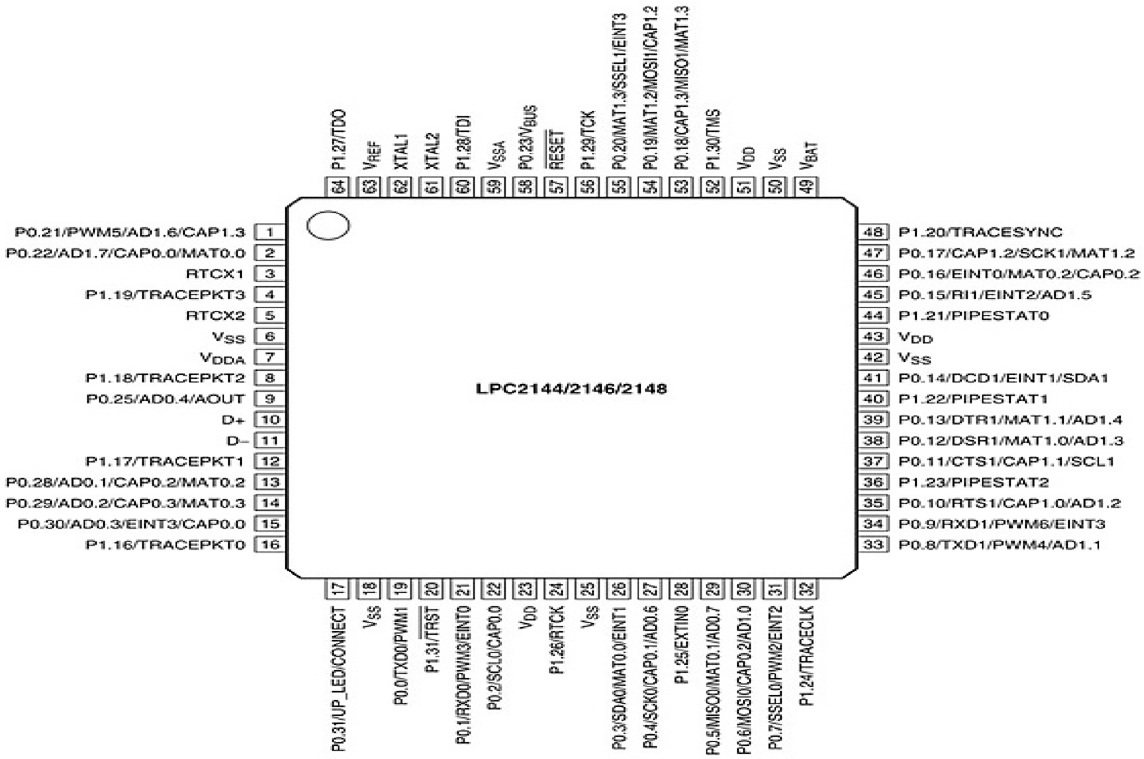


Figure 2.3: pin diagram of Arm7

2.3.2 Functional block diagram of Arm7 microcontroller

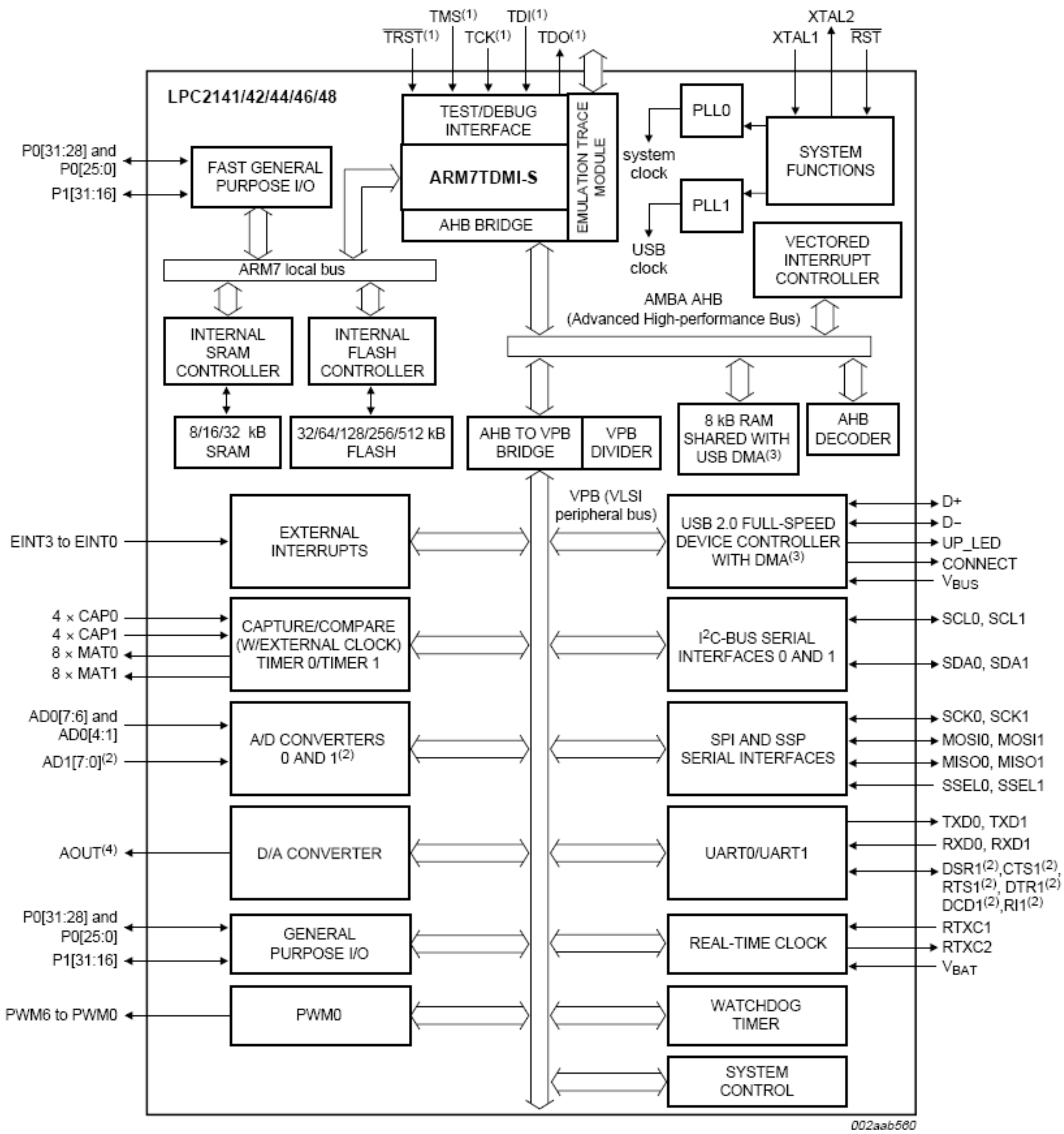


Figure 2.4: block diagram of LPC2148

2.3.3 Pin Description of microcontroller LPC2148

TheLPC2148 processor has totally four ports.

1. Port0 has 32 pins and all can be used as input/output. All pins of this port can be used as general purpose input/output. The number of pins available for input/output operation will depends on the use of alternate functions i.e. if we use less alternate functions more are the available input/output's. Port Pins P0.24,P0.26,P0.27 are not available.
2. Port1 has16 pins and all can be used as input/output. All pins of this port can be used as general purpose input/output. This is same as port0, only difference is this port has only 16pins where as port0 has 32 pins.

2.3.3.1 GPIO

Features

1. GPIO will give the direction control (whether the selected pin is used as input pin or output pin) of individual bits. It can be achieved by IODIR.

2. We can set the values of register by writing one, produces high at the corresponding port pins, whereas writing zero will have no effect. It can be done by using IOSET. Whenever we use i.e. when we set any value we have to clear those bit using IOCLR. IOCLR will clear the particular bits we have selected.

3. After reset, by default all the I/O will act as input pins.

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

Table 2.1: GPIO Pin Description.

Register Description

As we seen in the above table it is clear that LPC2148 has two 32-bit general purpose input/output ports. For Port0 29 pins (24, 26, 27 are not available) out of 32 pins are available for GPIO functions and for port1 only 16 (0-15pins are not available) out of 32 are available for GPIO functions. Port0 and port1 are controlled by two groups of four registers (IOPIN, IOSET, IODIR and IOCLR) which are explained in detail below.

There are four registers associated with the GPIO and are shown below

Generic Name	Description	Access	Reset Value	PORT0 Address & Name	PORT1 Address & Name
IOPIN	GPIO Port Pin Value Register. The current status of the GPIO configured port pins can always be read from this register, regardless of pin direction and mode.	Read Only	NA	0xE0028000 IO0PIN	0xE0028010 IO1PIN
IOSET	GPIO Port Output Set Register. This register controls the state of output pins along with the IOCLR register. Writing 1 produces highs at the corresponding port pins. Writing zeros has no effect.	Read/ Write	0x0000 0000	0xE0028004 IO0PIN	0xE0028014 IO1SET
IODIR	GPIO Port Direction Control Register. This register is used to control the direction of each port pin.	Read/ Write	0x0000 0000	0xE0028008 IO0DIR	0xE0028018 IO1DIR
IOCLR	GPIO Port Output Clear Register. This register is used to control the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeros has no effect.	Write Only	0x0000 0000	0xE0028008 IO0DIR	0xE0028014 IO1SET

Table 2.2: GPIO Register Description

2.3.3.2 GPIO Port Pin Value Register:

This is a 32 bit register. In our project we are not using this register as we are selecting the particular port pins as outputs. Here we know the output pins. This register is used when we would like to know whether the particular port pins used as output or not.

IOPIN	Description	Value after Reset
31:0	GPIO pin value bits. Bit0 in IOPIN corresponds to P0.0.... Bit 31 in IOPIN corresponds to P0.31.	undefined

Table2.3: GPIO Port PIN Value Register (IOPIN Register).

2.3.3.3 GPIO Port Output Set Register:

GPIO Output Set Register is a 32 bit register used to make the particular bits to high level output at the port pins if they are configured as GPIO in an output mode. Writing 1 makes a high level at the particular port pins, whereas writing 0 will have no effect. If any pin is configured as input then writing to IOSET has no effect.

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit0 in IOSET corresponds to P0.0.... Bit 31 in IOSET corresponds to P0.31.	0

Table 2.4: GPIO Port Output Set Register (IOSET Register).

2.3.3.4 GPIO Port Direction Register:

GPIO Direction Register is a 32 bit register used to control the direction of the pins whether the port pins used as input or output. If we write 1 then the corresponding port pin is selected or used as output. Direction bit for any pin must be set according to the pin functionality.

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit0 in IOSET corresponds to P0.0.... Bit 31 in IOSET corresponds to P0.31.	0

Table 2.5: GPIO Port Direction Register (IODIR Register).

2.3.3.5 GPIO Port Output Clear Register:

GPIO Output Clear Register is a 32 bit register used to produce a low level at port pins if they are configured as GPIO in output mode. In this register writing 1 will produce low level at the corresponding port pins and clears the corresponding bits in the IOSET register, because once the bits are set using IOSET register, they must be made low by using IOCLR register. Writing 0 will have no effect.

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit0 in IOSET corresponds to P0.0.... Bit 31 in IOSET corresponds to P0.31.	0

Table2.6: GPIO Port Output Clear Register (IOCLR Register).

UART0

Features

1. It has 16 byte Transmit and Receive FIFO’s.

2. It has built-in baud rate generator.

3. UART0 Register locations are confirmed to 550 industry standards.

Pin Description

In LPC2148 we are having only one UART which is UART0. Generally RS-232 is used as the UART0. In Every UART input is to receive the data and output is to transmit the data i.e. Receiver we will receive the input data and transmitter will output the data. TXD pin of UART0 is connected to 8^{th} pin of port0 which is TDX1 of the processor and RXD pin of UART0 is connected to 9^{th} pin of port0 of the processor.

IOSET	**Description**	**Value after Reset**
31:0	Output value SET bits. Bit0 in IOSET corresponds to P0.0.... Bit 31 in IOSET corresponds to P0.31.	0

Table 2.7: UART0 PIN Description.

Register Description

UART0 of LPC2148 contains ten 8-bit registers **[1]**. All these registers are listed below:

1. UART0 Receive Buffer Register (U0RBR).
2. UART0 Transmitter Holding Register (U0THR).
3. UART0 Interrupt Enable Register (U0IER).
4. UART0 Interrupt Identification Register (U0IIR).
5. UART0 FIFO Control Register (U0FCR).
6. UART0 Line Control Register (U0LCR).
7. UART0 Line Status Register (U0LSR).
8. UART0 Scratch Pad Register (U0SCR).
9. UART0 Divisor Latch LSB (U0DLL).
10. UART0 Divisor Latch MSB (U0DLM).

The below figure shows all the registers of UART0 along with their bit description, their access (whether they are read only or write only or both read and write), their reset values and their address

Name	**Description**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**Access**	**Reset Value**	**Address**
U0RB	Receive Buffer Register	MSB READ DATA LSB								RO	Un-define	0xE000C000 DLAB = 0
U0THR	Transmit Holding Register	MSB WRITE DATA LSB								WO	NA	0xE000C000 DLAB = 0
U0IER	Interrupt Enable Register	0	0	0	0	0				R/W	0	0xE000C004 DLAB = 0
U0IIR	Interrupt ID Register	FIFO Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01	0xE000C008
U0FCR	FIFO Control Register	Rx Trigger		Reserved		-				WO	0	0xE000C008
U0LCR	Line Control Register						N	Word Length Select		R/W	0	0xE000C008
U0LSR	Line Status Register	Rx FIFO Err	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60	0xE000C008
U0SCR	Scratch Pad Register	MSB LSB								R/W	0	0xE000C008

U0DLL	Divisor Latch LSB	MSB LSB	R/W	0x01	0xE000C008
U0DLM	Divisor Latch MSB	MSB LSB	R/W	0	0xE000C008

Table2. 8: Registers of UART0.

Out of these ten register only U0RBR, U0THR, U0LCR, U0LSR U0DLL, andU0DLM are used. The role of these registers is discussed later in our project.

2.3.3.6 UART0 Receiver Buffer Register:

In order to access UART0 Receiver buffer register, firstly we have to make the Divisor Latch Access Bit (DLAB) in Line Control Register (U0LCR) to zero. The UART0RBR is always read only. We know that U0RBR is the top byte of the UART0 Rx FIFO. Here the top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface and the LSB represents the oldest received data bit. In our project we are using the characters which are less than 8-bits. If the character is less than 8-bits, the unused MSB's must me padded with zeros .

U0RBR	Function	Description	Reset Value
7:0	Receive Buffer Register	The UART0 Receive Buffer Register contains the oldest received byte in the UART0 Rx FIFO.	Undefined

Table 2.9: UART0 Receive Buffer Register (U0RBR).

2.3.3.7 UART0 Transmitter Holding Register:

In order to access UART0 Transmitter Holding Register, firstly we have to make the Divisor Latch Access Bit (DLAB) which is present in Line Control Register (U0LCR) to zero. The U0THR is always write only. We know that U0THR is the top byte of the UART0 Tx FIFO. Here the top byte is the newest character in the Tx FIFO and can be written via the bus interface. We know that the LSB represents the first bit to transmit. In our project we are keeping our command (values given to the processor in order to control the devices) in UART0 Transmitter Holding Register. If the data present in the UART0 THR matches with the predefined command, we can get control to monitor the devices on the board. We are placing the command in between “$A__@” to differentiate the next command with the previous command.

U0THR	Function	Description	Reset Value
7:0	Transmit Holding Register	Writing to the UART0 Transmit Holding Register causes the data to be stored in the UART0 Transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter s available.	Undefined

Table 2.10: UART0 Transmitter Holding Register (U0THR).

2.3.3.8 UART0 Line Control Register:

The UART0 Line Control Register determines the format of the data character that is to be transmitted or received. In our project U0LCR is used to get access to U0DLL, U0DLM, U0LCR and U0THR by using the DLAB bit. In our program we are using 8bit character length, 1 start bit with no parity. After setting the DLAB bit we can get access to set the baud rate. After setting the baud rate we have to disable this bit to keep the baud rate constant.

U0LCR	Function	Description	Reset value
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0:1 Stop bit 1:2 Stop bits (1.5 if U0LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

Table2.11: UART0 Line Control Register (U0LCR).

2.3.3.9 UART0 Line Status Register:

The UART0 Line Status Register will provide the status information which is present on the UART0 Tx and Rx. U0LSR is a read-only register. In our project we are using this register to check whether the data is received or not. When we send a character, the control will stay there till the character is received or when next character is pressed

2.3.3.10 UART0 Divisor Latch MSB Register:

When we are using UART0 Divisor Latches, the DLAB bit present in the U0LCR must be one. U0DLM along with U0DLL is a 16-bit divisor. In this 16-bit divisor U0DLL will occupy the lower 8-bits and U0DLM will have higher 8-bits of the divisor. The UART0 Divisor Latch is a part of UART0 baud rate generator. It will divide the VPB clock in order to produce the baud rate clock. Baud rate clock must be 16x the desired baud rate.

U0DLM	Function	Description	Reset value
7:0	Divisor Latch MSB Register	The UART0 Divisor Latch MSB Register along with U0DLL register determines the baud rate of the UART0.	Undefined

Table 2.12: UART0 Divisor Latch MSB Register (U0DLM).

2.3.3.11 UART0 Divisor Latch LSB Register:

DLAB bit in U0LCR register must be one in order to access this register. By using this register we can set the baud rate which is required. Generally we use baud rate of 9600.
This baud rate is achieved by passing 0x62 to this register.

CHAPTER 3

DESIGN APPROACH

In this chapter we discuss about the block diagram of Development of Gas Leak Detection and Location System Based on Wireless Technology hardware requirements and software requirements.

3.1 BLOCK DIAGRAM OF DEVELOPMENT OF GAS LEAK DETECTIONAND LOCATION SYSTEM BASED ON WIRELESS TECHNOLOGY

It contains transmitter and receiver sections the below figure shows the transmitter block diagram of Development of Gas Leak Detection and Location System Based on Wirless Technology

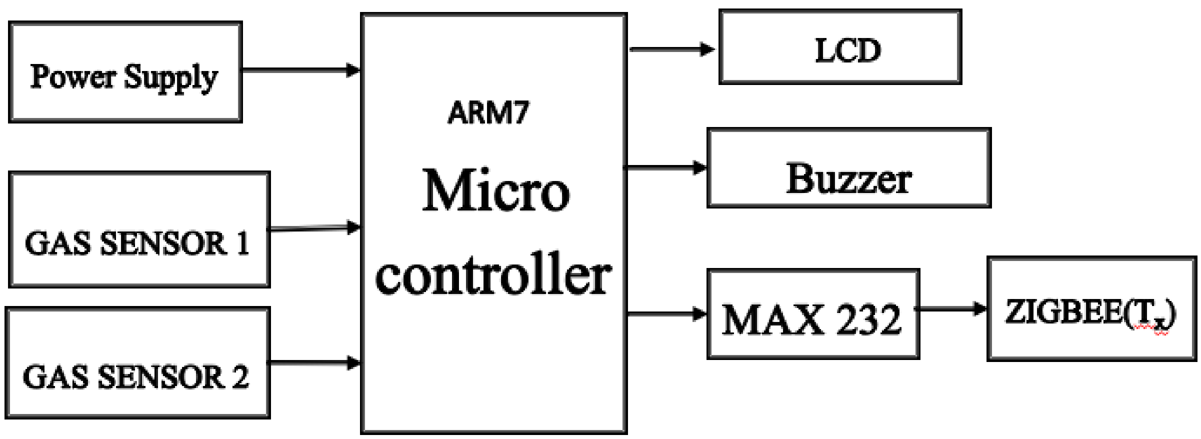


Figure 3.1: Transmitter (T_X) block Diagram of the Development of Gas Leak Detection and Location System Based on Wireless Technology

The below figure shows the Receiver(R_X) section of an Development of Gas Leak Detection and Location System Based on Wirless Technology

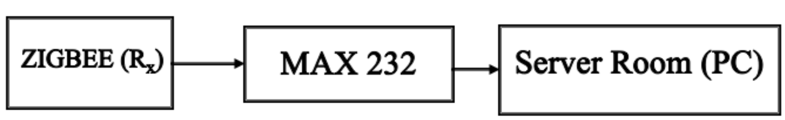
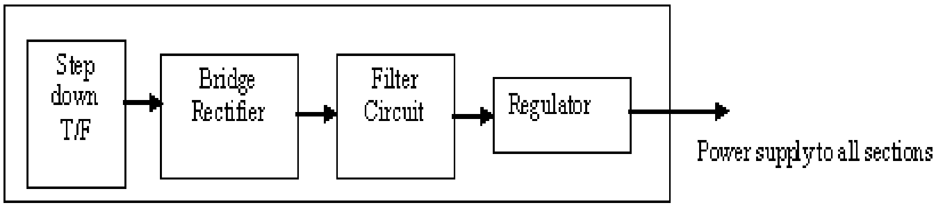


Figure 3.2:Receiver(R_X) block diagram of Development of Gas Leak Detection and Location System Based on Wirless Technology



3.2 Description of the Transmitter (T_X) and Receiver (R_X)block diagrams:-

The AC main Block is the power supply, which is of single-phase 230V ac. This should be give to step down transformer to reduce the 230V ac voltage to low voltage. i.e., to 6V or 12V ac this value depends on the transformer inner winding. The output of the transformer is give to the rectifier circuit. This rectifier converts ac voltage to dc voltage. Nevertheless, the voltage may consist of ripples or harmonics.

To avoid these ripples the output of the rectifier is connect to filter. The filter thus removes the harmonics. This is the exact dc voltage of the given specification. However, the circuit operates at 5V dc voltage. Therefore, we need a regulator to reduce the voltage. 7805 regulator produces 5V dc voltage.

This 5V D.C voltage is give to the all blocks present in the block diagram. In this project we use LPC2148 microcontroller. This is the heart of this project. Max232, ZIGBEE, LCD, , motor all are connected to this microcontroller.

TRANSMITTER CIRCUIT EXPLANATION: - In this project we required operating voltage for ARM controller board is 12V. Hence the 12V D.C. power supply is needed for the ARM board. This regulated 12V is generated by stepping down the voltage from 230V to 18V now the step downed a.c voltage is being rectified by the Bridge Rectifier using 1N4007 diodes. The rectified a.c voltage is now filtered using a ‘C’ filter. Now the rectified, filtered D.C. voltage is fed to the Voltage Regulator. This voltage regulator provides/allows us to have a Regulated constant Voltage which is of +12V. The rectified; filtered and regulated voltage is again filtered for ripples using an electrolytic capacitor 100μF. Now the output from this section is fed to microcontroller board to supply operating voltage.

RECEIVER CIRCUIT EXPLANATION: -In this project we required operating voltage for ARM controller board is 12V. Hence the 12V D.C. power supply is needed for the ARM board. This regulated 12V is generated by stepping down the voltage from 230V to 18V now the step downed a.c voltage is being rectified by the Bridge Rectifier using 1N4007 diodes. The rectified a.c voltage is now filtered using a 'C' filter. Now the rectified, filtered D.C. voltage is fed to the Voltage Regulator. This voltage regulator provides/allows us to have a Regulated constant Voltage which is of +12V. The rectified; filtered and regulated voltage is again filtered for ripples using an electrolytic capacitor 100μF. Now the output from this section is fed to microcontroller board to supply operating voltage

3.3 HARDWARE TOOLS

In this project the hardware requirements are following:

- Power supply unit
- Liquid Crystal Display
- MAX 232
- Transceiver
- Buzzer
- Gas Sensor

3.3.1 Power supply unit

The input to the circuit is applied from the regulated power supply. The a.c. input i.e., 230V from the mains supply is step down by the transformer to 12V and is fed to a rectifier. The output obtained from the rectifier is a pulsating d.c voltage. So in order to get a pure d.c voltage, the output voltage from the rectifier is fed to a filter to remove any a.c components present even after rectification. Now, this voltage is given to a voltage regulator to obtain a pure constant dc voltage.

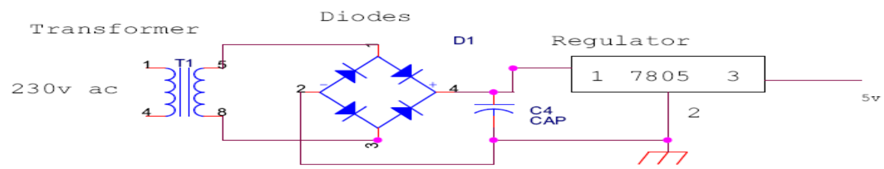


Figure 3.3: Circuit Diagram of Power Supply

Power Supply Unit Consists Of Following Units

- Step down transformer
- Rectifier unit
- Filter
- Regulator unit

3.3.1.1 Step-down Transformer

Usually, DC voltages are required to operate various electronic equipment and these voltages are 5V, 9V or 12V. But these voltages cannot be obtained directly. Thus the a.c input available at the mains supply i.e., 230V is to be brought down to the required voltage level. This is done by a transformer. Thus, a step down transformer is employed to decrease the voltage to a required level.

Figure 3.4: Step-down Transformers

3.3.1.2 Rectifier

The output from the transformer is fed to the rectifier. It converts A.C. into pulsating D.C. The rectifier may be a half wave or a full wave rectifier. In this project, a bridge rectifier is used because of its merits like good stability and full wave rectification.

The Bridge rectifier is a circuit, which converts an ac voltage to dc voltage using both half cycles of the input ac voltage. The Bridge rectifier circuit is shown in the figure.

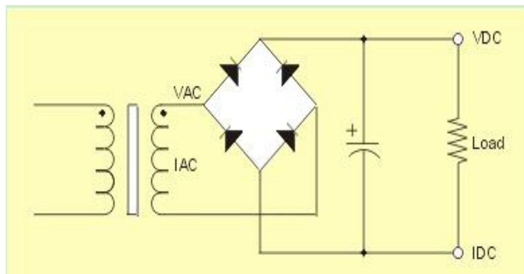


Figure 3.5: Bridge Wave Rectifier Circuit

For the positive half cycle of the input ac voltage, diodes D1 and D3 conduct, whereas diodes D2 and D4 remain in the OFF state. The conducting diodes will be in series with the load resistance R_L and hence the load current flows through R_L.

For the negative half cycle of the input ac voltage, diodes D2 and D4 conduct whereas, D1 and D3 remain OFF. The conducting diodes D2 and D4 will be in series with the load resistance R_L and hence the current flows through R_L in the same direction as in the previous half cycle. Thus a bi-directional wave is converted into a unidirectional wave.

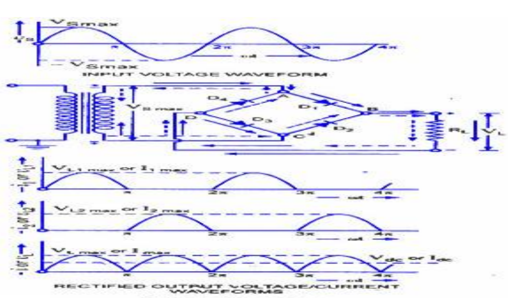


Figure 3.6: Output Voltage/Current Waveforms of Bridge Rectifier

3.3.1.3 Filter

Capacitive filter is used in this project. It removes the ripples from the output of rectifier and smoothens the D.C. Output received from this filter is constant until the mains voltage and load is maintained constant. However, if either of the two is varied, D.C. voltage received at this point changes. Therefore a regulator is applied at the output stage.

3.3.1.4 Voltage regulator

As the name itself implies, it regulates the input applied to it. A voltage regulator is an electrical regulator designed to automatically maintain a constant voltage level. In this project, power supply of 5V and 12V are required. In order to obtain these voltage levels, 7805 and 7812 voltage regulators are to be used. The first number 78 represents positive supply and the numbers 05, 12 represent the required output voltage levels. The L78xx series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3, D2PAK and DPAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1 A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

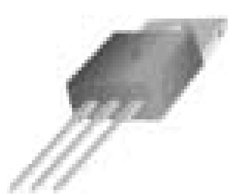


Figure 3.7: 7805 Regulators

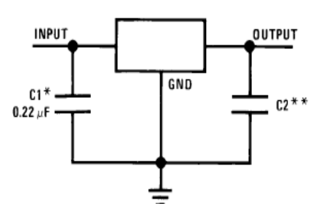


Figure 3.8: Regulators

3.3.2 Liquid Crystal Display

LEDs. It displays all thealphabets, Greek letters, punctuation marks, mathematical symbols etc

3.3.2.1 Pins Functions

There are the pins used for connection to the microcontroller.

Table3.1: LCD pins description

Function	Pin Number	Name	Logic State	Description
Ground	1	Vss	-	0V
Power supply	2	Vdd	-	+5V
Contrast	3	Vee	-	0 - Vdd
	4	RS	0 1	D0 – D7 are interpreted as commands D0 – D7 are interpreted as data
Control of operation	5	R/W	0 1	Write data (from controller to LCD) Read data (from LCD to controller)
	6	E	0 1 From 1 to 0	Access to LCD disabled Normal operating Data/commands are transferred to LCD
Data / commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1

	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6
	14	D7	0/1	Bit 7 MSB

3.3.2.2 LCD BASIC COMMANDS

All data transferred to LCD through outputs D0-D7 will be interpreted as commands or as data, which depends on logic state on pin RS:RS = 1 - Bits D0 - D7 are addresses of characters that should be displayed. Built in processor addresses built in “map of characters” and displays corresponding symbols. Displaying position is determined by DDRAM address. This address is either previously defined or the address of previously transferred character is automatically incremented

Table3.2: LCD basic commands

Command	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Execution Time
Clear display	0	0	0	0	0	0	0	0	0	1	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	x	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	40uS
Display on/off control	0	0	0	0	0	0	1	D	U	B	40uS
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x	40uS
Function set	0	0	0	0	1	DL	N	F	x	x	40uS
Set CGRAM address	0	0	0	1		CGRAM address					40uS
Set DDRAM address	0	0	1			DDRAM address					40uS
Read "BUSY" flag (BF)	0	1	BF			DDRAM address					-
Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	40uS
Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	40uS

I/D 1 = Increment (by 1)
0 = Decrement (by 1)

R/L 1 = Shift right
0 = Shift left

S 1 = Display shift on

DL 1 = 8-bit interface

	0 = Display shift off		0 = 4-bit interface
D	1 = Display on	N	1 = Display in two lines
	0 = Display off		0 = Display in one line
U	1 = Cursor on	F	1 = Character format 5x10 dots
	0 = Cursor off		0 = Character format 5x7 dots
B	1 = Cursor blink on	D/C	1 = Display shift
	0 = Cursor blink off		0 = Cursor shift

3.3.2.3 LCD CONNECTION

Depending on how many lines are used for connection to the microcontroller, there are 8-bit and 4-bit LCD modes. The appropriate mode is determined at the beginning of the process in a phase called “initialization”. In the first case, the data are transferred through outputs D0-D7 as it has been already explained. In case of 4-bit LED mode, for the sake of saving valuable I/O pins of the microcontroller, there are only 4 higher bits (D4-D7) used for communication, while other may be left unconnected.

Consequently, each data is sent to LCD in two steps: four higher bits are sent first (that normally would be sent through lines D4-D7), four lower bits are sent afterwards. With the help of initialization, LCD will correctly connect and interpret each data received. Besides, with regards to the fact that data are rarely read from LCD (data mainly are transferred from microcontroller to LCD) one more I/O pin may be saved by simple connecting R/W pin to the Ground. Such saving has its price. Even though message displaying will be normally performed, it will not be possible to read from busy flag since it is not possible to read from display.

3.3.2.4 LCD INITIALIZATION

Once the power supply is turned on, LCD is automatically cleared. This process lasts for approximately 15mS. After that, display is ready to operate. The mode of operating is set by default. This means that:

1. Display is cleared

2. Mode

   DL = 1 Communication through 8-bit interface

   N = 0 Messages are displayed in one line

   F = 0 Character font 5 x 8 dots

3. Display/Cursor on/off

   D = 0 Display off

   U = 0 Cursor off

   B = 0 Cursor blink off

4. Character entry

   ID = 1 Addresses on display are automatically incremented by 1

   S = 0 Display shift off

Automatic reset is mainly performed without any problems. Mainly but not always! If for any reason power supply voltage does not reach full value in the course of 10mS, display will start perform completely unpredictable? If voltage supply unit can not meet this condition or if it is needed to provide completely safe operating, the process of initialization by which a new reset enabling display to operate normally must be applied.

Contrast control

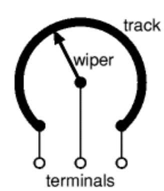


Figure 3.9: Variable resistors

To have a clear view of the characters on the LCD, contrast should be adjusted. To adjust the contrast, the voltage should be varied. For this, a preset is used which can behave like a variable voltage device. As the voltage of this preset is varied, the contrast of the LCD can be adjusted.

Potentiometer

Variable resistors used as potentiometers have all three terminals connected. This arrangement is normally used to vary voltage, for example to set the switching point of a circuit with a sensor, or control the volume (loudness) in an amplifier circuit. If the terminals at the ends of the track are connected across the power supply, then the wiper terminal will provide a voltage which can be varied from zero up to the maximum of the supply.

Presets

These are miniature versions of the standard variable resistor. They are designed to be mounted directly onto the circuit board and adjusted only when the circuit is built. For example to set the frequency of an alarm tone or the sensitivity of a light-sensitive circuit. A small screwdriver or similar tool is required to adjust presets. Presets are much cheaper than standard variable resistors so they are sometimes used in projects where a standard variable resistor would normally be used.

3.3.3 Max232

A standard serial interface for PC, RS232C, requires negative logic, i.e., logic 1 is -3V to -12V and logic 0 is +3V to +12V. To convert TTL logic, say, TxD and RxD pins of the microcontroller thus need a converter chip. A MAX232 chip has long been using in many microcontrollers boards. It is a dual RS232 receiver / transmitter that meets all RS232 specifications while using only +5V power supply. It has two onboard charge pump voltage converters which generate +10V to -10V power supplies from a single 5V supply. It has four level translators, two of which are RS232 transmitters that convert TTL/CMOS input levels into +9V RS232 outputs. The other two level translators are RS232 receivers that convert RS232 input to 5V. Typical MAX232 circuit is shown below.

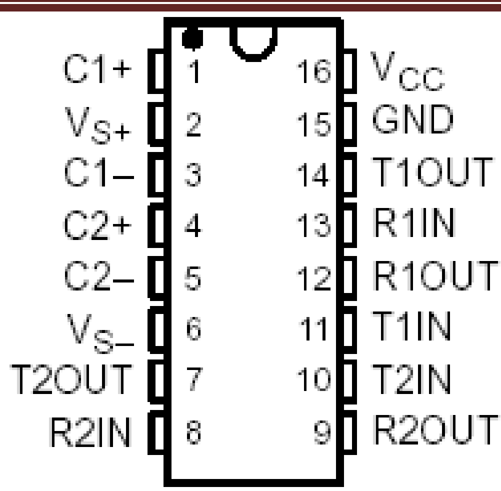


Figure 3.10: Pin diagram of MAX 232

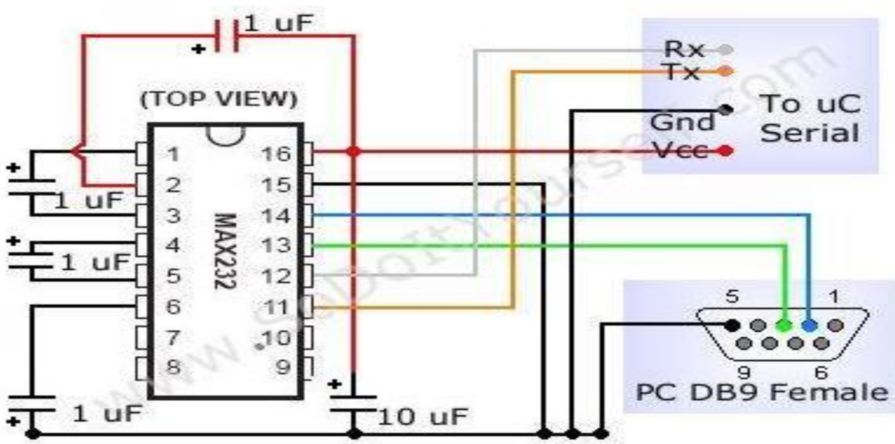
Figure 3.11: MAX232

Features:

1. Operates With Single 5-V Power Supply
2.LinBiCMOSE Process Technology
3.Two Drivers and Two Receivers
4.±30-V Input Levels
5.Low Supply Current . 8 mA Typical
6.Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
7.Designed to be Interchangeable With Maxim MAX232
8.ApplicationsTIA/EIA-232-F Battery-Powered SystemsTerminals Modems Computers
9.ESD Protection Exceeds 2000 V Per MIL-STD-883, Method 3015
10.Package Options Include Plastic Small-Outline (D, DW) Packages and Standard Plastic (N) DIPs

Circuit connections:

A standard serial interfacing for PC, RS232C, requires negative logic, i.e., logic '1' is -3V to -12V and logic '0' is +3V to +12V. To convert a TTL logic, say, TxD and RxD pins of the uC chips, thus need a converter chip. A MAX232 chip has long been using in many uC boards. It provides 2-channel RS232C port and requires external 10uF pacitors. Carefully check the polarity of capacitor when soldering the board. A DS275 however, no need external capacitor and smaller. Either circuit can be used without any problems.



3.3.4 TRANSCEIVER

Pin Diagram

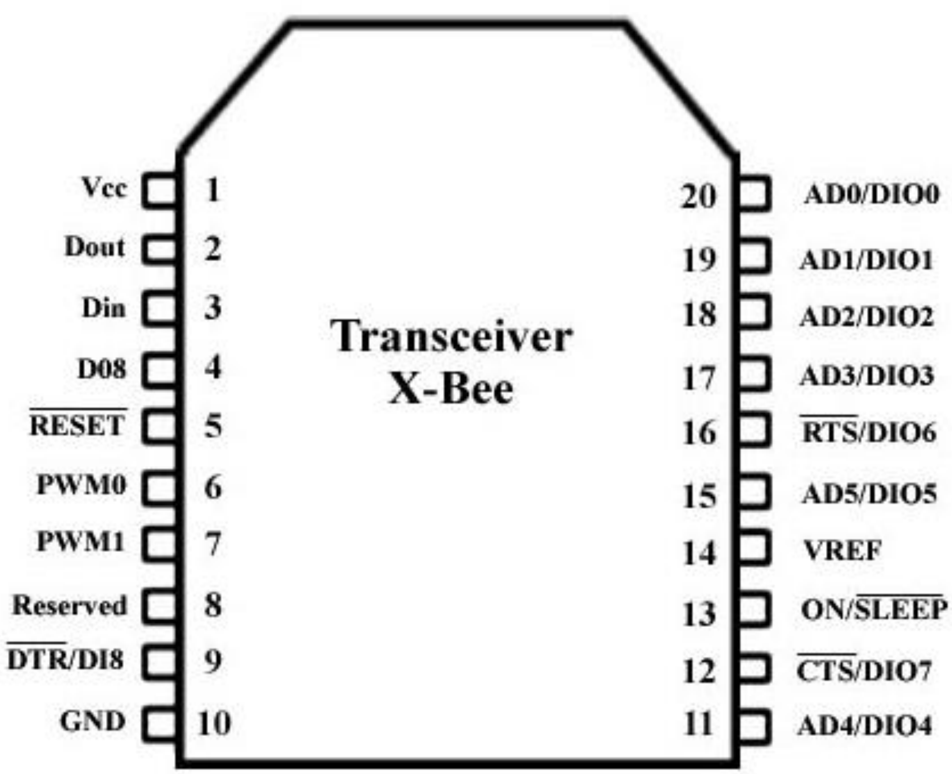


Figure 3.12: Pin diagram of X-Bee Transceiver

Zigbee modules feature a UART interface, which allows any microcontroller or microprocessor to immediately use the services of the Zigbee protocol.The logic level conversion can be performed using either a standard RS-232 IC or logic level translators such as the 74LVTH125 when the host is directly connected to the XBee UART. The below table gives the pin description of transceiver.

Table 3.3: Pin Description of X-Bee Transceiver

Pin	Name	Direction	Description
1	Vcc	-	Power Supply
2	DOUT	Output	UART Data Out
3	DIN/CONFIG	Input	UART Data In
4	DO8	Output	Digital Output 8
5	RESET	Input	Module Reset
6	PWM0/RSSI	Output	PWM Output 0/RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DDR/SLEEP_RQ/DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4/DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS/DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON/SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate/AD5/DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS/AD6/DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3/DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2/DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1/DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0/DIO0	Either	Analog Input 0 or Digital I/O 0

Design Notes:

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS and DTR
- Signal Direction is specified with respect to the module
- Module includes a 50kΩ pull-up resistor attached to RESET
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

Performance:

Table 3.4: Performance characteristics

Parameters	Value
Indoor/Urban Range	30m
Outdoor RF (LOS)	100m
Transmit Power Output	1mW (0dBm)
RF Data Rate	250,000bps
Serial Interface Data Rate	1200-115200bps
Receiver Sensitivity	-92dBm

Power Requirements:

Table 3.5:Power Requirement characteristics

Parameters	Value
Supply Voltage	2.8 - 3.4V
Transmit Current	45mA
Receive Current	50mA

General:

Table 3.6: General characteristics

Parameters	Value
Operating Frequency	ISM 2.4GHz
Dimensions	2.468 x 2.761
Operating Temperature	-40^{o} to 85^{o} C
Antenna Options	Integrated Chip Antenna

Networking and Security:

Table 3.7: Networking and Security characteristics

Parameters	Value
Supported Network Topologies	Point-to-point, Point-to-multipoint, Peer-to-peer
Number of Channels	16 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses

6.3 System Data Flow Diagram

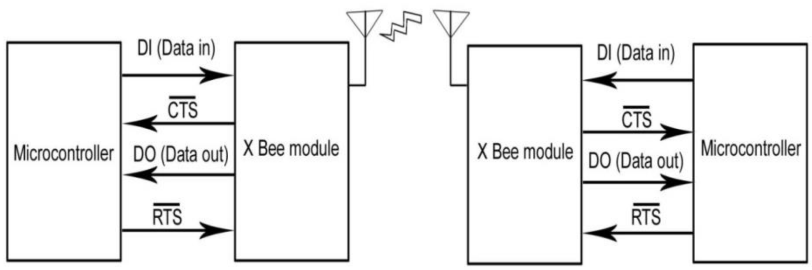


Figure 3.13: Data Flow Diagram

The X-Bee RF Modules interface to a host device through a logic-level asynchronousSerial port. Through its serial port, the module can communicate with any logic and voltageCompatible UART; or through a level translator to any serial device.

Data is presented to the X-Bee module through its DIN pin, and it must be in the asynchronous serial format, which consists of a start bit, 8 data bits, and a stop bit. Because the input data goes directly into the input of a UART within the X-Bee module, no bit inversions are necessary within the asynchronous serial data stream. All of the required timing and parity checking is automatically taken care of by the X-Bee's UART.

Just in case you are producing data faster than the X-Bee can process and transmit it, both X-Bee modules incorporate a clear-to-send (CTS) function to throttle the data being presented to the X-Bee module's DIN pin. You can eliminate the need for the CTS signal by sending small data packets at slower data rates.

If the microcontroller wants to send data to transceiver, it will send RTS (Request to Send) signal. If the transceiver is idle it sends CTS (Clear to Send) signal. The RTS and CTS signals are active low. When microcontroller receives CTS command it will send data to the transceiver through DIN pin. The transceiver will send the data to microcontroller through DOUT pin. The communication between transceiver and the microcontroller at the transmitter and receiver is similar. The communication between transmitter and receiver is through RF communication.

3.3.5 BUZZER

Figure 3.14: Buzzer

A **buzzer** or **beeper** is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers and confirmation of user input such as a mouse click or keystroke.

Mechanical

A joy buzzer is an example of a purely mechanical buzzer.

Electromechanical

Early devices were based on an electromechanical system identical to an electric bell without the metal gong. Similarly, a relay may be connected to interrupt its own actuating current, causing the contacts to buzz. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

Piezoelectric

A piezoelectric element may be driven by an oscillating electronic circuit or other audio signal source, driven with a piezoelectric audio amplifier. Sounds commonly used to indicate that a button has been pressed are a click, a ring or a beep.

Uses

- Annunciator panels
- Electronic metronomes
- Game shows
- Microwave ovens and other household appliances
- Sporting events such as basketball games
- Electrical alarms

3.3.6 GAS SENSOR

Figure 3.15: Gas Sensor

Gas sensors are available in wide specifications depending on the sensitivity levels, type of gas to be sensed, physical dimensions and numerous other factors. This Insight covers a **methane gas sensor** that can sense gases such as ammonia which might get produced from methane. When a gas interacts with this sensor, it is first ionized into its constituents and is then adsorbed by the sensing element. This adsorption creates a potential difference on the element which is conveyed to the processor unit through output pins in form of current. What is this sensing element? Is it kept in some chamber or is kept exposed? How does it get current and how it is taken out? Let’s find out in this Insight!

3.4 SOFTWARE REQUIREMENTS

The software’s used for this project are:

3.4.1 EMBEDDED C

HI-TECH Software makes industrial-strength software development tools and C compilers that help software developers write compact, efficient embedded processor code.

For over two decades HI-TECH Software has delivered the industry's most reliable embedded software development tools and compilers for writing efficient and compact code to run on the most popular embedded processors. Used by tens of thousands of customers including General Motors, Whirlpool, Qualcomm, John Deere and many others, HI-TECH's reliable development tools and C compilers, combined with world-class support have helped serious embedded software programmers to create hundreds of breakthrough new solutions.

Whichever embedded processor family you are targeting with your software, whether it is the ARM, PICC or 8051 series, HI-TECH tools and C compilers can help you write better code and bring it to market faster.

HI-TECH PICC is a high-performance C compiler for the Microchip PIC micro 10/12/14/16/17 series of microcontrollers. HI-TECH PICC is an industrial-strength ANSI C compiler - not a subset implementation like some other PIC compilers. The PICC compiler implements full ISO/ANSI C, with the exception of recursion. All data types are supported including 24 and 32 bit IEEE standard floating point. HI-TECH PICC makes full use of specific PIC features and using an intelligent optimizer, can generate high-quality code easily rivaling hand-written assembler. Automatic handling of page and bank selection frees the programmer from the trivial details of assembler code.

3.4.1.1 EMBEDDED C COMPILER

- ANSI C - full featured and portable
- Reliable - mature, field-proven technology
- Multiple C optimization levels

- Full linker, with overlaying of local variables to minimize RAM usage
- Comprehensive C library with all source code provided
- Includes support for 24-bit and 32-bit IEEE floating point and 32-bit long data types
- Mixed C and assembler programming
- Listings showing generated assembler
- Compatible - integrates into the MPLABhttp://www.htsoft.com/company/trademarks.php IDE, MPLAB ICD and most 3rd-party development tools

3.4.1.2 EMBEDDED SYSTEM TOOLS

ASSEMBLER

An assembler is a computer program for translating assembly language — essentially, a mnemonic representation of machine language — into object code. A cross assembler (see cross compiler) produces code for one type of processor, but runs on another. The computational step where an assembler is run is known as assembly time. Translating assembly instruction mnemonics into opcodes, assemblers provide the ability to use symbolic names for memory locations (saving tedious calculations and manually updating addresses when a program is slightly modified), and macro facilities for performing textual substitution — typically used to encode common short sequences of instructions to run inline instead of in a subroutine. Assemblers are far simpler to write than compilers for high-level languages.

Assembly language has several benefits:

- **Speed**: Assembly language programs are generally the fastest programs around.
- **Space**: Assembly language programs are often the smallest.
- **Capability:** You can do things in assembly which are difficult or impossible in High level languages.

- **Knowledge**: Your knowledge of assembly language will help you write better programs, even when using High level languages. An example of an assembler we use in our project is RAD 51.

SIMULATOR

Simulator is a machine that simulates an environment for the purpose of training or research.

3.4.2 KEIL SOFTWARE

It is possible to create the source files in a text editor such as Notepad, run the Compiler on each C source file, specifying a list of controls, run the Assembler on each Assembler source file, specifying another list of controls, run either the Library Manager or Linker (again specifying a list of controls) and finally running the Object-HEX Converter to convert the Linker output file to an Intel Hex File. Once that has been completed the Hex File can be downloaded to the target hardware and debugged. Alternatively KEIL can be used to create source files; automatically compile, link and covert using options set with an easy to use user interface and finally simulate or perform debugging on the hardware with access to C variables and memory. Unless you have to use the tolls on the command line, the choice is clear. KEIL Greatly simplifies the process of creating and testing an embedded application.

Projects

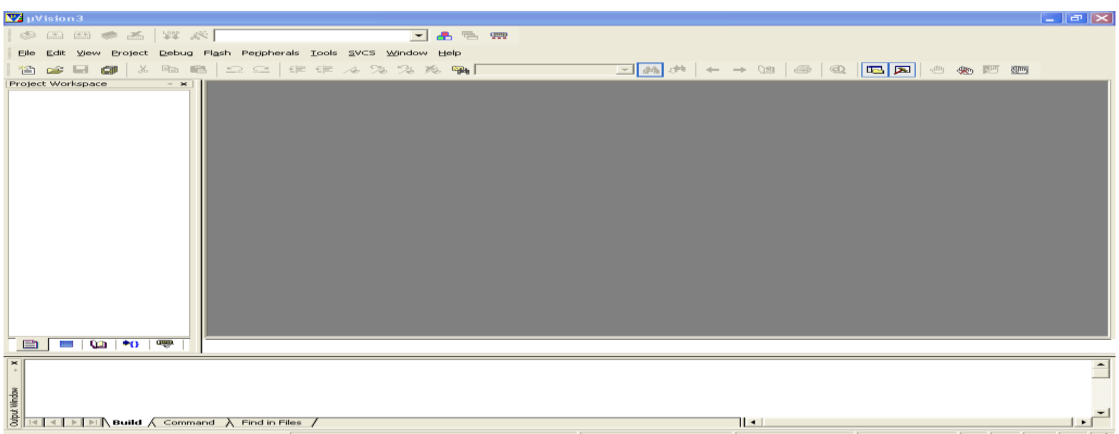
The user of KEIL centers on “projects”. A project is a list of all the source files required to build a single application, all the tool options which specify exactly how to build the application, and – if required – how the application should be simulated. A project contains enough information to take a set of source files and generate exactly the binary code required for the application. Because of the high degree of flexibility required from the tools, there are many options that can be set to configure the tools to operate in a specific manner. It would be tedious to have to set these options up every time the application is being built; therefore they are stored in a project file. Loading the project file into KEIL informs KEIL which source files are required, where they are, and how to configure the tools in the correct way. KEIL can then execute each tool with the correct options. It is also possible to create new projects in KEIL. Source files are added to the project and the tool options are set as required. The project can then be saved to preserve the settings. The project is reloaded and the simulator or debugger started, all the desired windows are opened. KEIL project files have the extension

Simulator/Debugger:

The simulator/ debugger in KEIL can perform a very detailed simulation of a micro controller along with external signals. It is possible to view the precise execution time of a single assembly instruction, or a single line of C code, all the way up to the entire application, simply by entering the crystal frequency. A window can be opened for each peripheral on the device, showing the state of the peripheral. This enables quick trouble shooting of mis-configured peripherals. Breakpoints may be set on either assembly instructions or lines of C code, and execution may be stepped through one instruction or C line at a time. The contents of all the memory areas may be viewed along with ability to find specific variables. In addition the registers may be viewed allowing a detailed view of what the microcontroller is doing at any point in time.

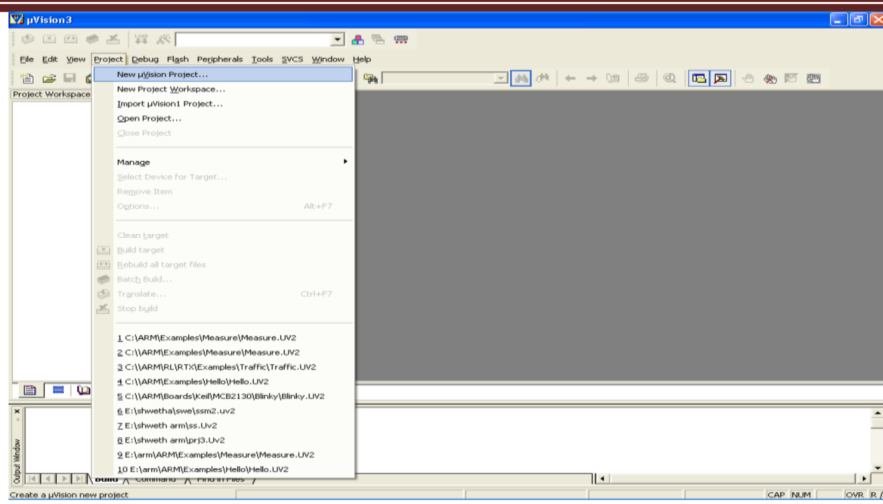
ABOUT KEILARM:

1. Click on the Keil u Vision3 Icon on Desktop
2. The following fig will appear

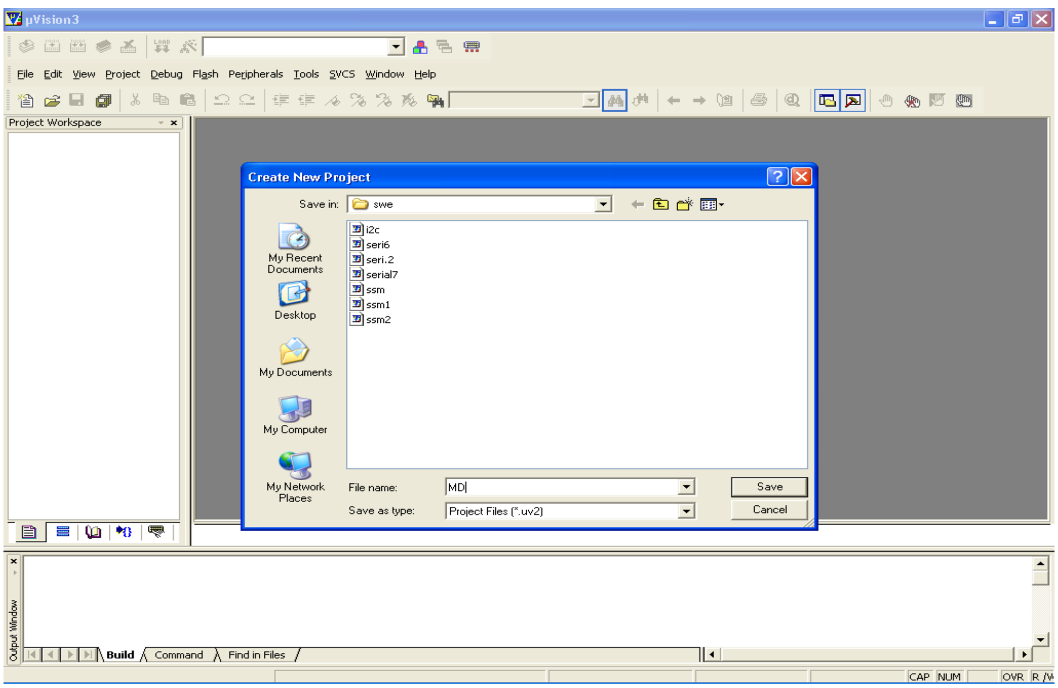


3.Click on the Project menu from the title bar

4.Then Click on New Project



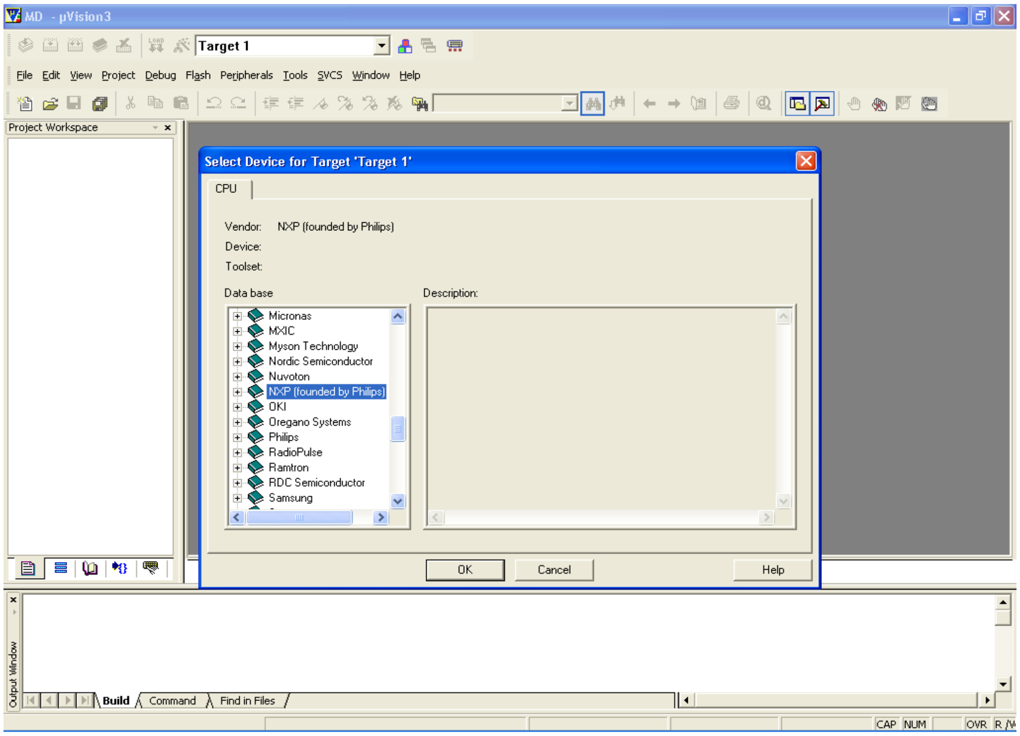
5.Save the Project by typing suitable project name with no extension in u r own folder sited in either C:\ or D:\



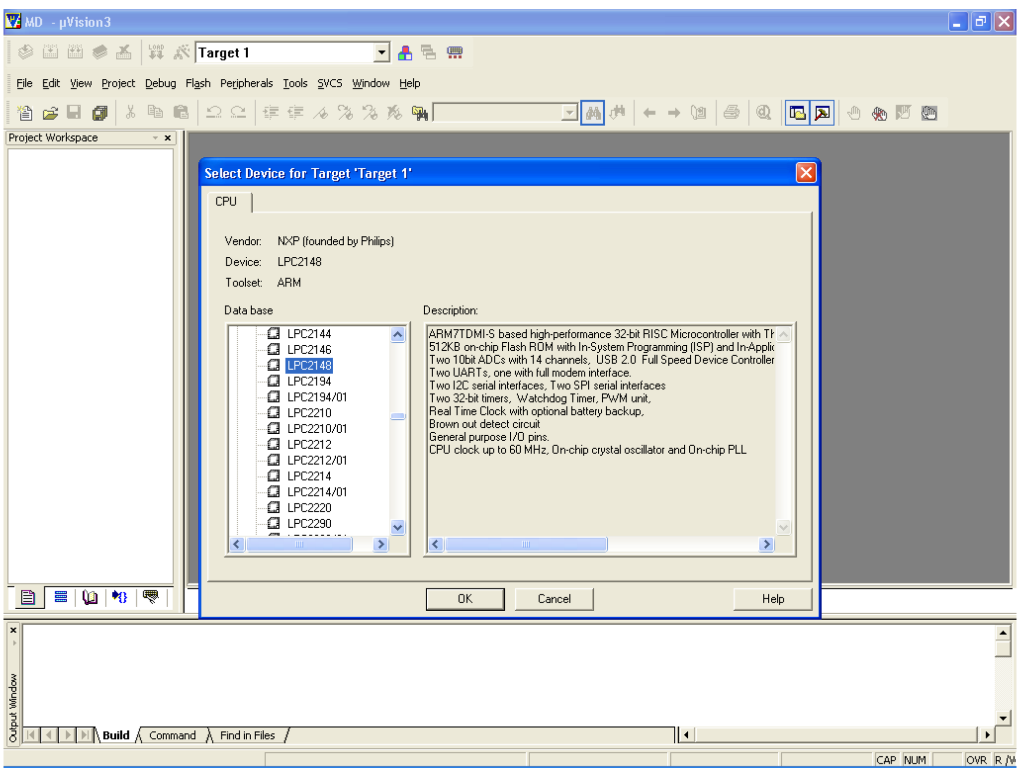
6.Then Click on Save button above.

7.Select the component for u r project. i.e.NXP……

8.Click on the + Symbol beside of NXP

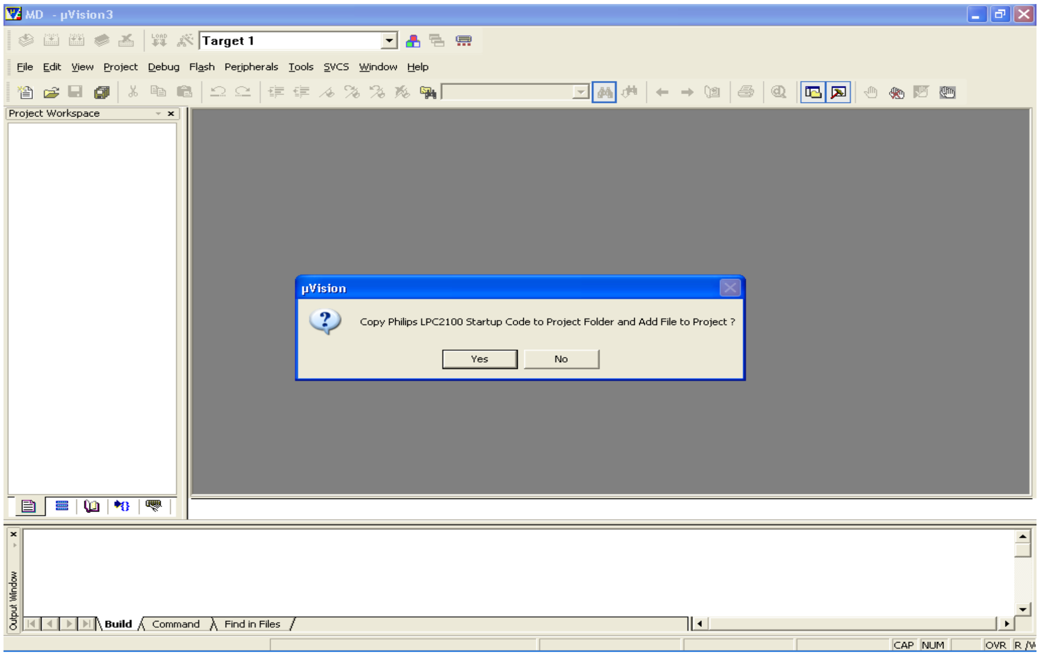


9.Select LPC2148 as shown below



10.Then Click on “OK”

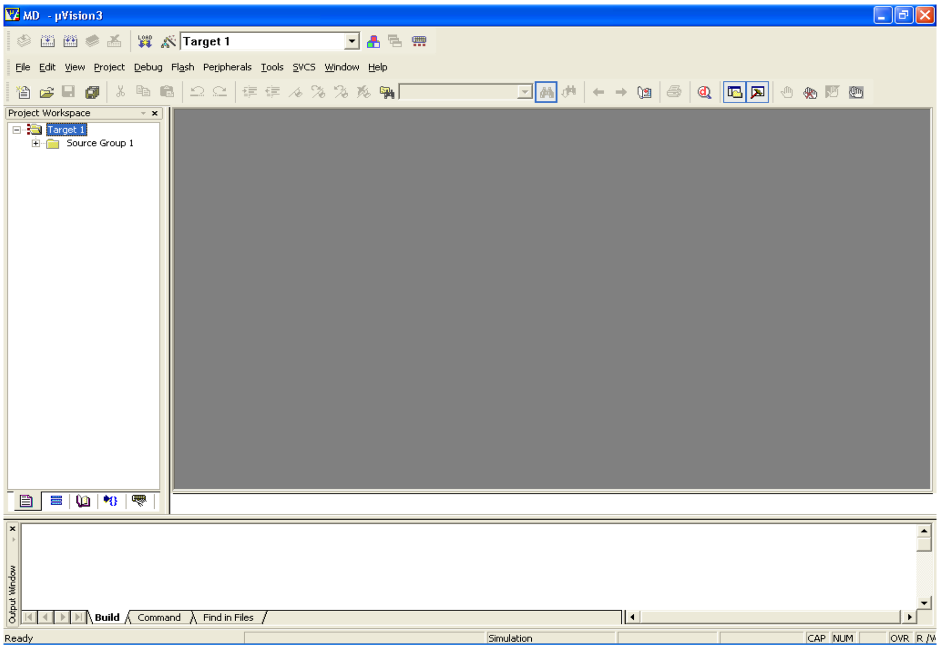
11.The Following fig will appear



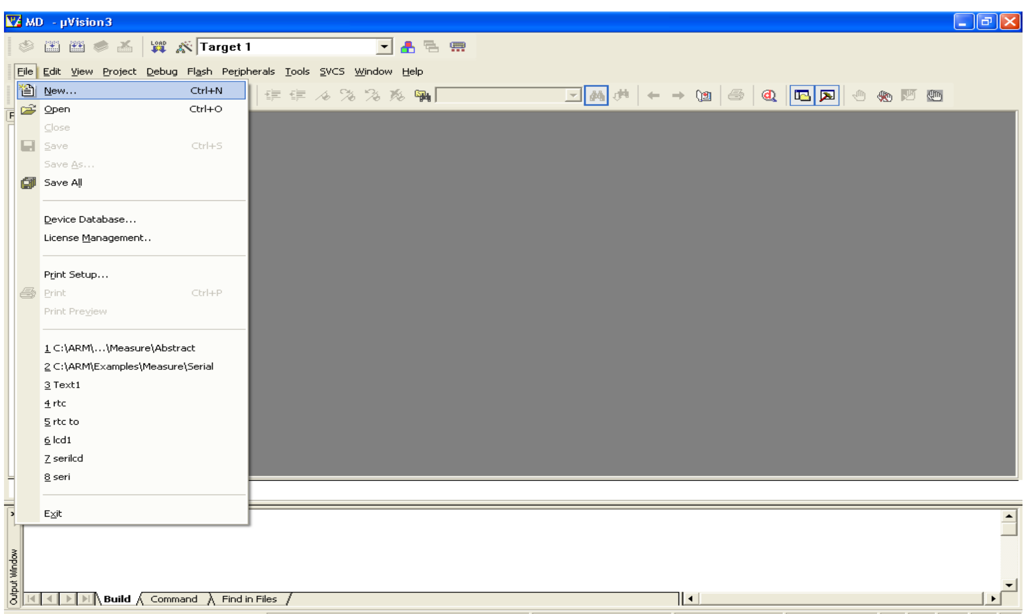
12.Then Click YES

13.Now your project is ready to USE

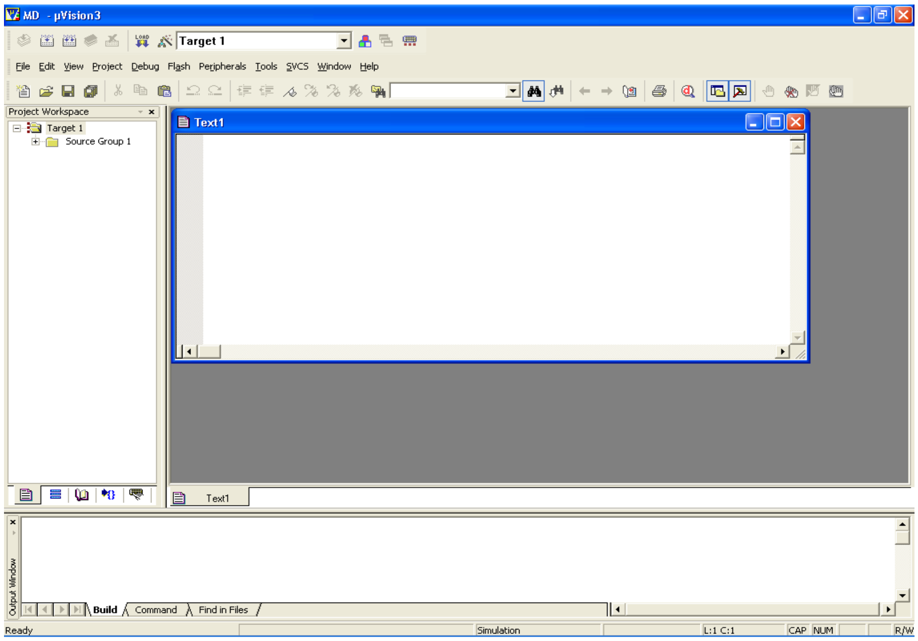
14.Now double click on the Target1, you would get another option “Source group 1” as shown in next page.



15.Click on the file option from menu bar and select “new”

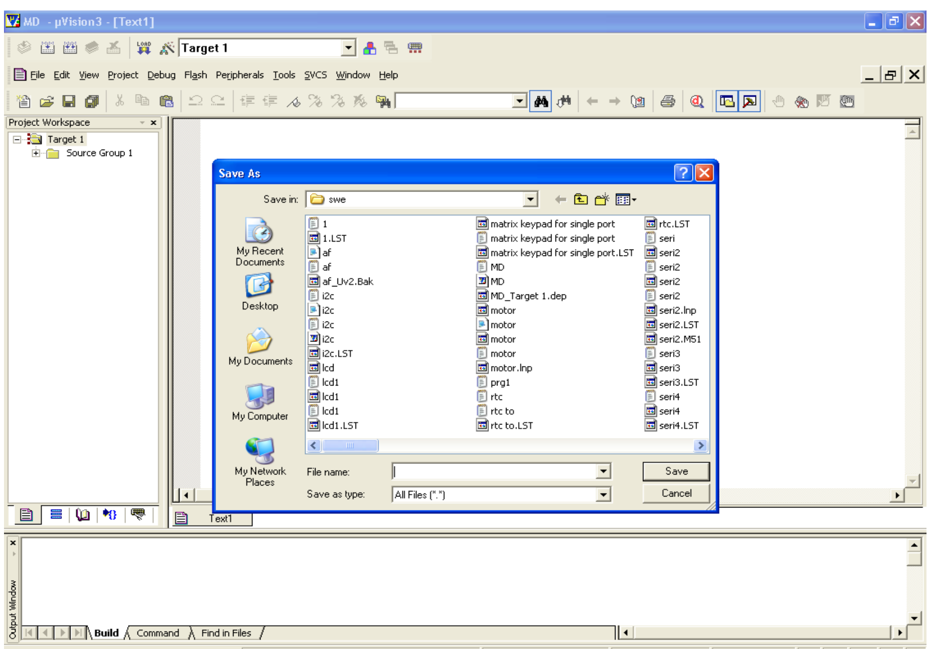


16.The next screen will be as shown in next page, and just maximize it by double clicking on its blue boarder.

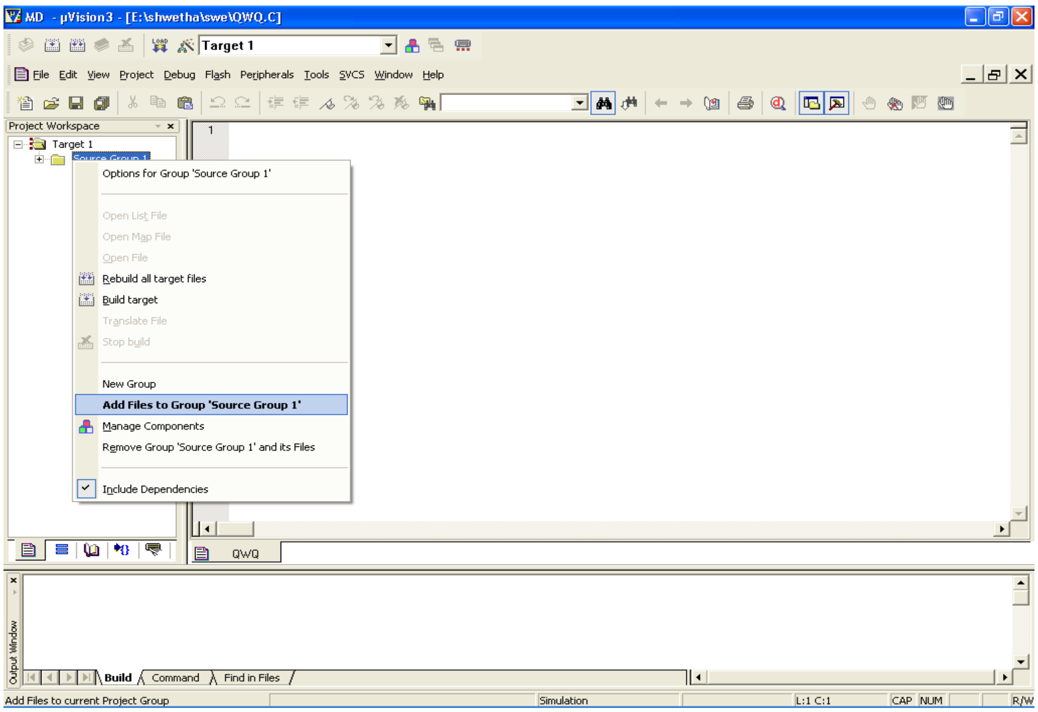


17.Now start writing program in either in “C” or “ASM”

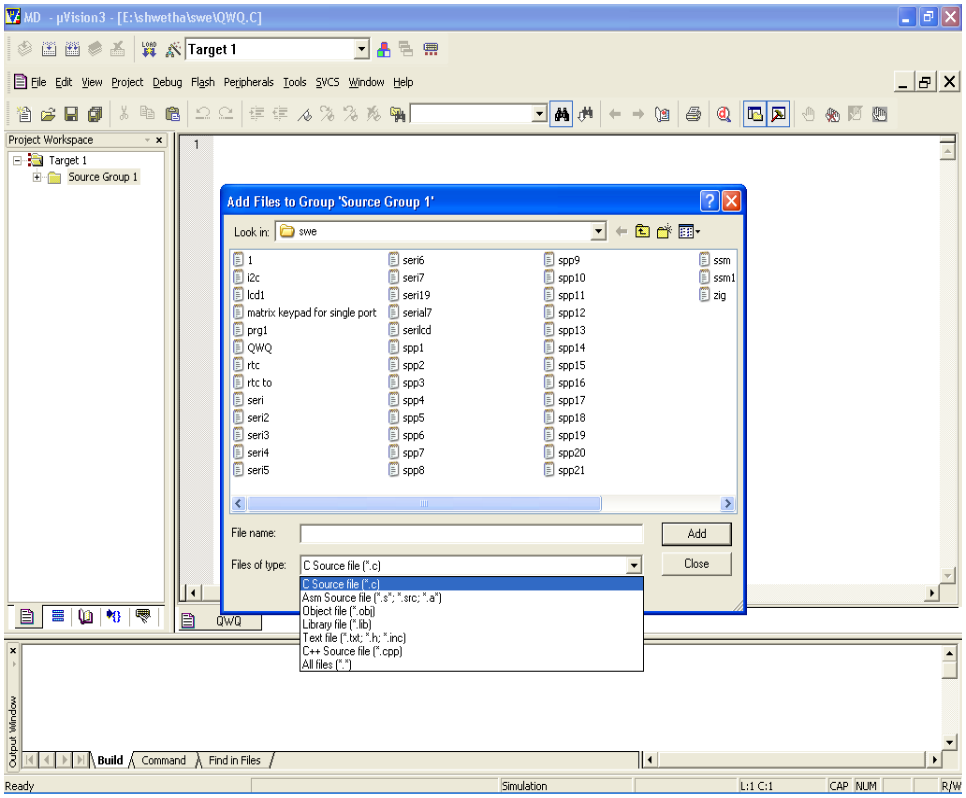
18.For a program written in Assembly, then save it with extension “. asm” and for “C” based program save it with extension “ .C”



19.Now right click on Source group 1 and click on “**Add files to Group Source**”



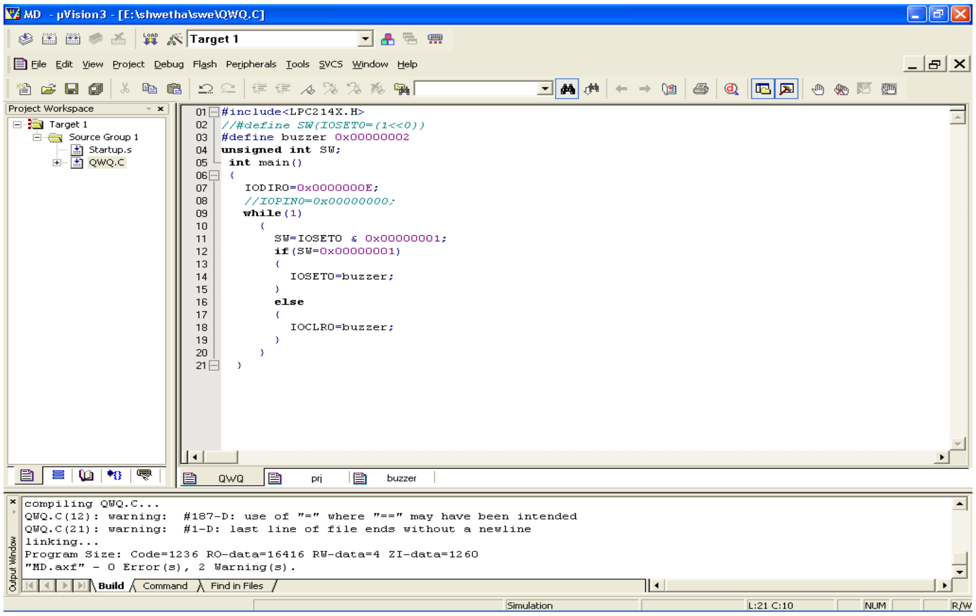
20.Now you will get another window, on which by default "C" files will appear



21.Now select as per your file extension given while saving the file

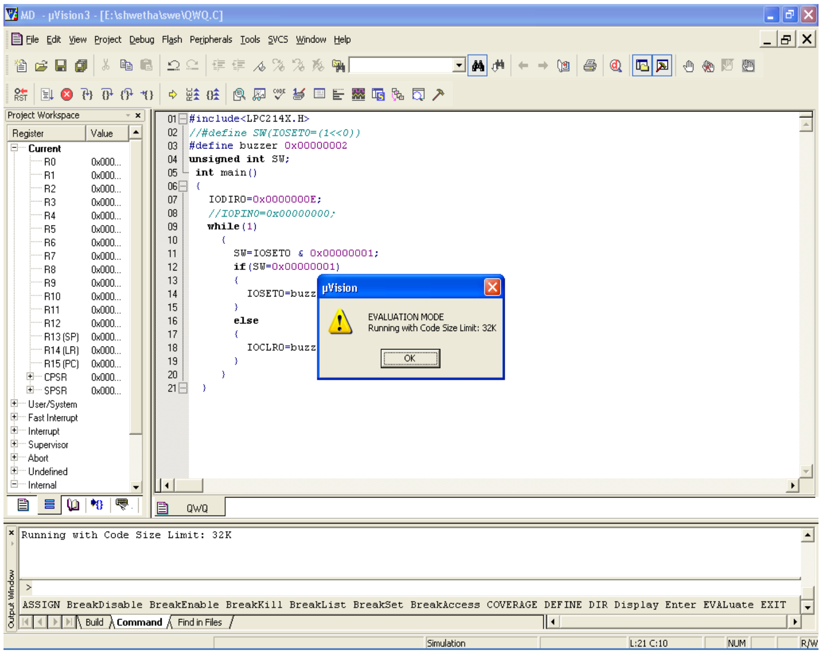
22.Click only one time on option “**ADD**”

23.Now Press function key F7 to compile. Any error will appear if so happen.



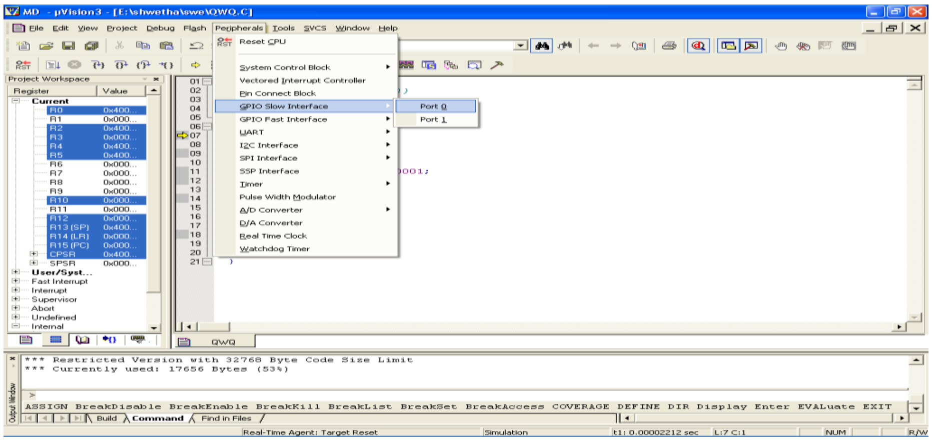
24.If the file contains no error, then press Control+F5 simultaneously.

25.The new window is as follows

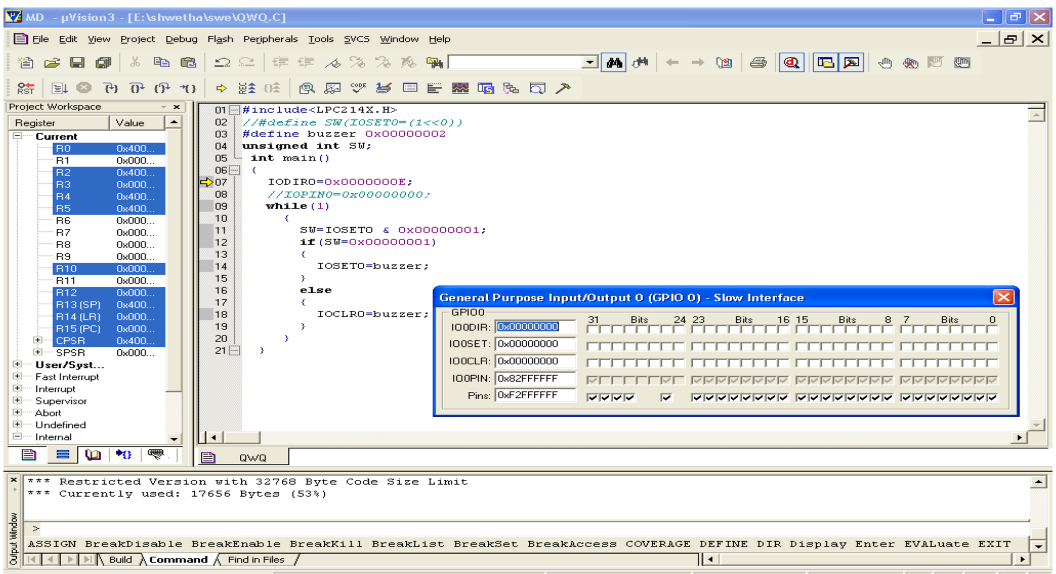


26.Then Click "OK"

27.Now Click on the Peripherals from menu bar, and check your required port as shown in fig below



29.Drag the port a side and click in the program file



29. Now keep Pressing function key “F11” slowly and observe.

30. You are running your program successfully

CHAPTER 4

RESULT

In chapter 4 we discuss about Result, Applications and advantages

4.1 Results of Development of Gas Leak Detection and Location System Based on Wirless Technology

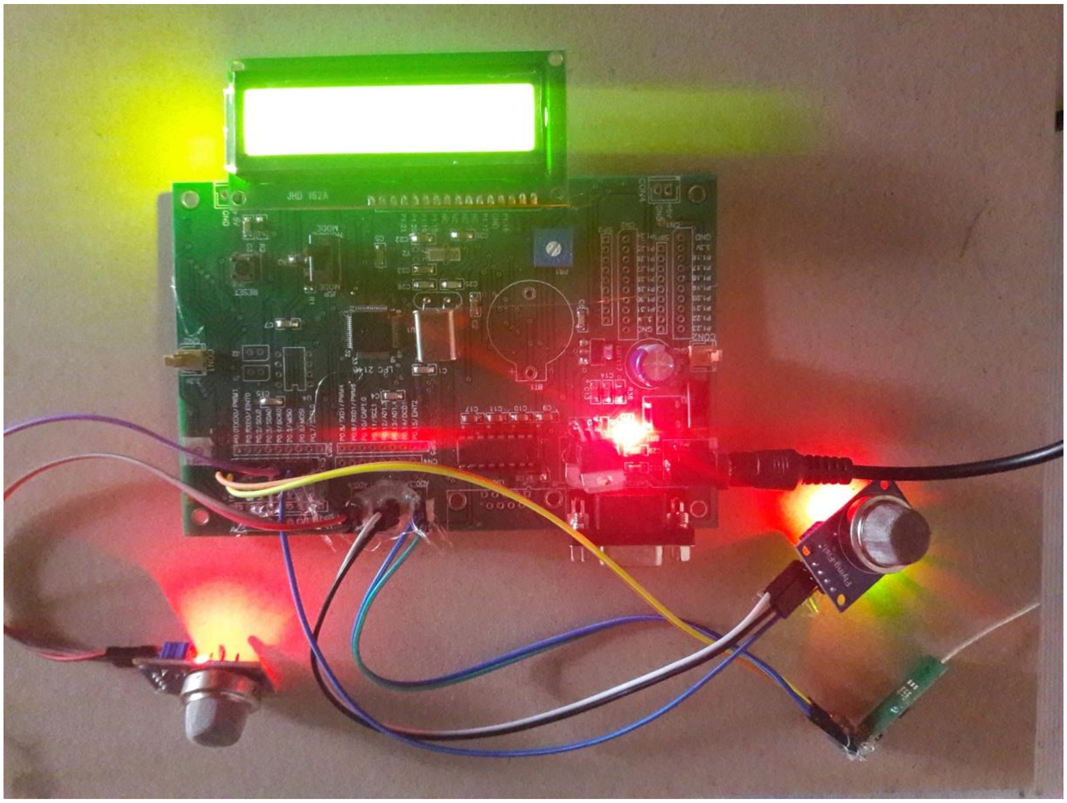


Figure 4.1: Development of Gas Leak Detection and Location System Based on Wirless Technologykit image

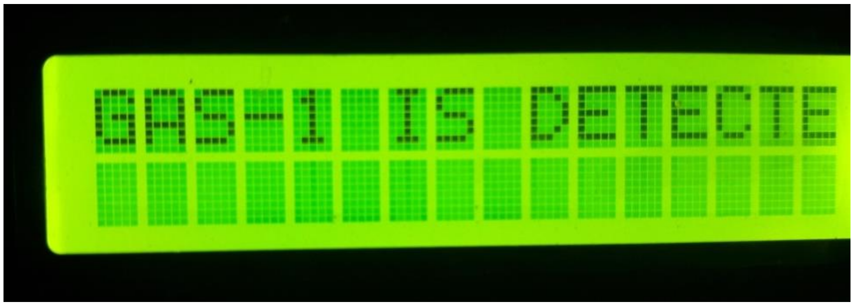


Figure 4.2: Development of Gas Leak Detection and Location System Based on Wirless Technologykit image(Display)

Working

This project consists of ZIGBEE transmitter section and ZIGBEE receiver section. In ZIGBEE transmitter section it consists of sensor unit, display unit and controller unit. The sensor detects the gas leak and sends the information to the receiver controller unit PC and it display on PC where the gas leak. In ZIGBEE transmitter section it consists of Alarm unit, controller unit, and display unit. If the sensor in the transmitter section sends the information to the controller and it immediately display where the gas leak and alarm will be on that particular unit .This device will protect the person using the kitchen from indulging a fire accident as well as it do safety measure on gas leak automatically and safely.

First these kits are connected to the main supply (230V A.C).then it is step down to 5V d.c supply. 230V A.C supply is given as input to the step down transformer then it is step down that voltage to some 18V A.C supply. Then it is given to the Bridge wave Rectifier. This converts A.C to Pulsating D.C. then this is given to the filter circuit. Here capacitive filter is used. So it converts that pulsating D.C to pure D.C. next this is connected to 7805 regulator. It produces our required 5V D.C supply.

4.2ADVANTAGES

- Simple and Reliable
- Low Cost
- Easily applicable

4.3 APPLICATIONS

- It is applicable in restaurants, hotels.
- It is applicable in gas filling stations.
- It is applicable in gas operated vehicles

CHAPTER 5

CODING

```
#include <LPC214X.H>
#include "lcd.h"
#include  "uart.h"

main()
{
	VPBDIV=0X02;					//set the clock speed for 30 mhz
	PINSEL0=0X00000005;				//selecting uart o port pins
	IODIR1 |= 0X003F0000;		//calling direction for lcd pins
	uartini();				//calling uart inilization function
	delay(50);
	lcdini();					//calling lcd inilization function
	delay(50);
	cmd(0x80);
	lcdstr("GAS  LEAK DETECTION AND");
	cmd(0xc0);
	lcdstr("LOCATION SYSTEMS");
	cmd(0x01);				//lcd clear command
	delay(100);
	while(1)
	{
		if(!((IOPIN0)&(1<<6)))
		{
			cmd(0x01);
			lcdstr("GAS-1 IS DETECTED");
			sendstring("GAS-1 IS DETECTED\r\n");
		}
		else if(!(( IOPIN0)&(1<<7)))
		{
			cmd(0x01);
			lcdstr("GAS-2 IS DETECTED");
			sendstring("GAS-2 IS DETECTED\r\n");
		}
		else
		{
			cmd(0x01);
			lcdstr("GAS-1 AND GAS-2");
			cmd(0xc0);
			lcdstr(" NOT  DETECTED");
			sendstring("NOT DETECTED\r\n");
		}
	}
}


#define LCD 0X003C0000
#define RS  0X00010000
#define EN  0X00020000

void lcdini(void);
```

```
void lcd_display(unsigned char x);
void cmd(unsigned char x);
void lcdstr(unsigned char *str);
void delay(int count);

void lcdini()   // lcd initilization function
{
        cmd(0X02);  // to initialize LCD in 4-bit mode.
        cmd(0X28);  //to initialize LCD in 2 lines, 5X7 dots and 4bit mode.
        cmd(0x0e);  // cursor ON
        cmd(0X06);
        cmd(0X01);  // clear lcd
        cmd(0X80);  // cursor indicating first line first position
}

//-------------------------------------------------------------------

void lcd_display(unsigned char x)  // lcd display funtion
{
        unsigned int temp;    //initilize variable
        delay(700);        // calling delay
        IOCLR1 |= RS;
        IOCLR1 |= EN;
        IOCLR1 |= LCD;
        temp=(x>>4)&0x0f;   // rotating value of x by 4 and anding with 0x0f
        IOSET1 |=(temp<<18);  //put value of temp at on lcd pins
        IOSET1 |=RS;    // set re pin
        IOSET1 |=EN;    // set enable pin
        delay(10); // hault for some time
        IOCLR1 |=EN;   // clear enable pin

        delay(1000); // calling delay function
        IOCLR1 |= RS;
        IOCLR1 |= EN;
        IOCLR1 |= LCD;
        temp=x&0x0f;    // anding value of x with 0x0f
        IOSET1 |=(temp<<18); // putting value of temp on lcd data pin
        IOSET1 |=RS;  // set re pin
        IOSET1 |=EN;  // set enable pin
        delay(10); // hault for some time
        IOCLR1 |=EN;  // clear enable pin
        delay(100); // calling delay function
}
//--------------------------------------------------------------------
void cmd(unsigned char x)   // lcd command funtion
{
        unsigned int temp;      //initilize variable for msb
        IOCLR1 |= RS;
        IOCLR1 |= EN;
        IOCLR1 |= LCD;
```

```
        temp=(x>>4)&0x0f;   // rotating value of x by 4 and anding with 0x0f
        IOSET1 |=(temp<<18);  //put value of temp at on lcd pins
        IOCLR1 |=RS;  // clear re pin
        IOSET1 |=EN;  // set enable pin
        delay(100); // hault for some time
        IOCLR1 |=EN;  // clear enable pin

        delay(100);  // calling delay function
        IOCLR1 |= RS;
        IOCLR1 |= EN;
        IOCLR1 |= LCD;
        temp=x&0x0f;     // anding value of x with 0x0f
        IOSET1 |=(temp<<18);  // putting value of temp on lcd data pin
        IOCLR1 |=RS;  // clear re pin
        IOSET1 |=EN;  // set enable pin
        delay(100); // hault for some time
        IOCLR1 |=EN; // clear enable pin
        delay(100);           // calling delay function
}
//------------------------------------------------------------------
void lcdstr(unsigned char *str)  // funtion to write sting on lcd
{
        while(*str!='\0')  // check str for NULL
        {
                lcd_display(*str);  // write one characture from string
                str++;     // increament string
        }
}
//---------------------------------------------------------------------
void delay(int count)
{
        int j=0,i=0;
        for(j=0;j<count;j++)
        {
                for(i=0;i<500;i++);
        }
}
//------------------------------------------------------------------------------------------



void uartini(void);
void transmituart0(unsigned char val);
unsigned char receiveuart0(void);
void sendstring(unsigned char *str);

void uartini()
{

U0LCR=0x83; //8-N-        1, enable divisors
```

```
U0DLL=0xC3; //9600 baud (9615)
U0DLM=0x00;
U0LCR=0x03; //8-N-1, disable divisors
}
//-----------------------------------------------------------------------
void transmituart0(unsigned char val)                          //sends a byte through Uart
{
        while(!(U0LSR & 0x20));                      // checking TXD Buffer Empty
        U0THR =val;                          // Write Character
}
//---------------------------------------------------------------------------
unsigned char receiveuart0(void)
{
        while(!(U0LSR & 0x01));                   // Wait RXD Receive Data Ready
        return(U0RBR);                                     // Get Receice Data & Return
}

//---------------------------------------------------------------------------
void sendstring(unsigned char *str)                   //Sends a string of data through UART1
{
 unsigned int i=0;
 while(str[i]!='\0')
 {
   transmituart0(str[i]);
   i++;
 }
}
```

CHAPTER 6

CONCLUSION

A promising area of VANET, safety applications are attractingmore and more consideration. Monitoring and detectingthe behaviour of drivers is vital to ensuring road safety byalerting the driver and other vehicles on the road in cases ofabnormal driving behaviours. Driver behaviour is affected bymany factors that are related to the driver, the vehicle and theenvironment and over the course of driving a driver will befound to be in a particular state, the driver can then stay inthis state for a period of time or shift to another state. Hence,it is important to capture the static and the dynamic aspectsof behaviour and take into account the contextual informationthat relates to driver behaviour. In this paper we have presenteda driver behaviour detection system in VANET from the viewpoint of context awareness.

6.1 FUTURE ENHANCEMENT

The evaluation result has demonstrated the detection accuracy of the proposed model under uncertainty and the importance of including a great amount of contextual information within the inference process. Our future work comprises designing a corrective action algorithm to calculate the appropriate corrective actions for other vehicles on the road. Modeling techniques for transferring the data collected from sensors into a machine processable format will also be developed.

APPENDIX

REFERENCE

[1].JoaqunGutirrez, Juan Francisco Villa-Medina, AlejandraNieto- Garibay, And Miguel ngelPorta-Gndara, "AutomatedIrrigation System Using A Wireless Sensor Network And Gprs Module",Ieee Transactions On Instrumentation And Measurement, Vol. 63, No.1, January2014.

[2].Vimal.P, Priyanka.V, Rajyasree.M,SanthiyaDevi.P.T, Jagadeeshraja.M, SuthanthiraVanitha.N,"A Novel Approachfor Automatic Irrigation and Fertigation Using EmbeddedSystem," International Journal of VLSI and Embedded Systems-IjvesVol05, Article 03257; March2014.

[3].Sathiyabama P, Lakshmi Priya C, Ramesh Sm, PreethiB, Mohanaarasi M, "Embedded System Design For IrrigatingField With Different Crops Using Soil Moisture Sensor,"International Journal Of Innovative Research In Computer AndCommunication Engineering Vol. 2, Issue 8, August2014.

[4].LiaiGao, Meng Zhang, GengChen,"An Intelligent IrrigationSystem Based On Wireless Sensor Network and Fuzzy Control, "Journalof Networks, Vol. 8, No. 5, May2013.

[5].K.Prathyusha, M. ChaitanyaSuman,"Design of EmbeddedSystems fortheAutomationofDripIrrigation," International Journal of Application or Innovation in EngineeringManagement (Ijaiem) Volume 1, Issue 2, October2012.

[6].Orazio Mirabella, Senior Member, IEEE, and Michele Brischetto,"A Hybrid Wired/Wireless Networking Infrastructure For Greenhouse Management," IEEE Transactions on InstrumentationandMeasurement,Vol.60,No.2,February2011.

[7].B.Sivakumar, P.Gunasekaran, T.Selvaprabhu,P.Kumaran, D.Anandan, “The Application of Wireless Sensor Networkin the Irrigation Area Automatic System”, IjctaJan-Feb2012.

[8]. S.MuhammadUmair, R. Usman,”Automation Of IrrigationSystem Using Ann Based Controller,” International Journal OfElectrical Computer Sciences Ijecs-Ijens Vol: 10No: 02.May2010.

Web sites

[1].http://www.garmin.com/products/gps35

[2].http://www.alldatasheet.com

[3].http://www.mathworks.com

[4].M. A. Mazidi, J. C. Mazidi, R. D. Mckinaly, The 8051 Microcontroller and Embedded Systems, Pearson Education, 2006.

[5].http://www.national.com/ds/LM/LM35.pdf

[6]. http://www.nxp.com/documents/user_manual/UM10139.pdf