**GAS LEAKAGE MONITORING AND ALERTING**

**SYSTEM FOR INDUSTRIES**

**PROJECT REPORT**

**IBM-PROJECT-13063-1659509349**

**TEAM ID : PNT2022TMID08488**

**Submitted by**

**RUBASHREE S 810419106052**

**RAMANIKA R 810419106045**

**KEERTHIKA R 810419106023**

**VIDHYA V 810419106068**

**In partial fulfillment for the award of the degree of**

**BACHELOR OF ENGINEERING**

**IN**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE**

**(AUTONOMOUS)**

**Abstract**

Nowadays gas leakage and gas detection is a major problem in our daily lives. Also gas wastage is a major issue that needs to be countered. LPG gas is highly flammable and can inflict damage to life and property. To avoid such situations, a considerable amount of effort has been devoted to the development of reliable techniques for detecting gas leakage. As knowing about the existence of a leak is not always enough to launch a corrective action, some of the leak detection techniques were designed to allow the possibility of locating the leak. Our aim is to reduce the risks in Kitchen using Internet of Things. The main aim is to propose the design and construction of an SMS based Gas Leakage Alert System. Gas sensor are used to detect gas leakages in a kitchen. With the help of an infrared sensor the issue of gas wastage is also monitored. An alarm goes off whenever the sensor doesn’t detect any vessel over the burner beyond a particular time period

Introduction

Liquefied petroleum gas (LPG) is currently the most used gas in our home for cooking purposes. LPG gas is a flammable gas, if leaked it can cause major damage to life and property. Therefore it should be used in safe handling manner and additional care has to be taken in order to prevent any leakage possible. The main features of LPG is that being heavier than air, it do not disperse easily and may lead to suffocation when inhaled. The leaked gases when ignited may lead to explosion. The number of deaths due to the explosion of gas cylinders has been increasing in recent years. Now a days people are having very busy schedule and hence sometimes they forget or don’t get enough time for booking the gas from the gas agency. So it would be much easier and helpful if there was a provision to book the gas automatically. A major amount of gas is being wasted due to the carelessness of consumer’s .Sometimes they forget to turn off the burner which may also could lead to damages. Our proposed topic aims at detection of gas leakage and automatic controlling of gas valve. The smart gas system which provides home safety, detects the leakage of the LPG and alerts the consumer about the leak by a notification through by using android app through Internet Of Things (IOT) and consumer can turn off the gas valve , from anywhere in the world. The additional advantage of the system is that it continuously monitors the level of the LPG present in the cylinder using load sensor and if the gas level reaches below the threshold limit of gas so that the user can replace the old cylinder with new in time and books the cylinder by automatically send a notification to the gas agency. An added feature is that if the users accidently forget to turn off the gas burner, the system will inform by activating an alarm. so the problem of wastage of the energy is solved

**Internet of Things**

The ability of various things to be connected to each other through the Internet or It is network of physical devices (vehicles,building) connected to embedded device (software,sensor) through internet.IOT allows the object to sensor collect remotely across network of infrastructure. IOT contains various domains , protocols , application.The interconnection of these embedded devices is expected to usher in automation in nearly all fields, while also enabling advanced applications like a smart grid and expanding to the areas such as smart cities. At the same time, IOT is strongly tied to the big data era due to the enormous data that the “Things” can generate. For the interconnection of these devices, different wired or wireless standards exist. IOT provide various residential and enterprises solution through latest technology .It broadly covers M2M communication, smart grids, smart building, smart cities and many more application. Using IOT in smart cities/smart buildings can certainly provide reliable and efficient solutions as it will allow the user to interact with the entities.

**EXSITING SYSTEM**

Now a day’s every one want a facility which reduce their efforts, time and provide a way to do their work more easily. For cooking food we all uses LPG gass. LPG having versatile nature so its demand raise day by day. It mostly uses in domestic fuel, industrial fuel and automobile fuel. We find uneducated people are not able to do these task and busy schedule people they haven’t sufficient time to do all the activity. Also safety plays the important role. As we all know that many accident happens due to gas leakageknow the gas booking is time consuming task and also there can be problem of gas leakage . Most of the accident occur because of gas leakage. In these project we can identify the gas leakage, controlling it and also without any human interface gas booking is done



**Proposed system**

The proposed system consist of atmega328 and it is interfaced to sensors like MQ2 sensor, load cell, infrared sensors etc. which are the input of the system. Here the wifi-module is interfaced to these which give the ability to communicate with each other. The proposed method takes an automatic control action upon detection of gas.The regulator valve (motor) would be switched off which completely stops the flow of gas leakage. Initialy if there is a gas leakage then the electronic sensor i.e. the gas sensor that obeys the principle of LPG sensor senses any gas leakage from storage, if any leakage sensed then the output of this sensor goes high. This high signal is monitored by the microcontroller and it will identify the gas leakage. If there is a leakage, the consumer is informed through internet in his device and a signal is sent back to the microcontroller to turn off the valve. In this system, a sensor(load cell) is used to monitor the weight of the gas cylinder ,if it goes below a critical value the sensor senses this condition and sends a notification via internet to gas agency for booking a LPG. The object detection sensor is used to detect the presence of any vessel over the burner. If a vessel is not detected over a predetermined time, then an alarm goes off and the consumer is alerted.

**Block Diagram**

****

**Hardware specification**

**CHAPTER 2**

**LITERATURE SURVEY**

**[1] Soundarya, T., et al., C-leakage: Cylinder LPG Gas leakage Detection for home safety. IOSR Journal of Electronics and Communication (IOSR-JECE), 2014. 9(1): p. 53-58.**

Several passive microwave satellites orbit the Earth and measure rainfall. These measurements have the advantage of almost full global coverage when compared to surface rain gauges. However, these satellites have low temporal revisit and missing data over some regions. Image fusion is a useful technique to fill in the gaps of one image using another one. The proposed algorithm uses an iterative fusion scheme to integrate information from two satellite measurements. The algorithm is implemented on two datasets for 7 years of half-hourly data. The results show significant improvements in rain detection and rain intensity in the merged measurements. In order to produce a better estimation of the rainfall, researchers have tried to combine different types of rainfall measurement. Several models have been developed that combines satellite measurements with ground-base measurements, and provide estimation of rainfall for both missing pixels and for the times that there has not been a satellite measurement. The proposed algorithm has two sections: Texture Production and Shape Production. In order to produce the texture, it uses the multi-resolution pyramid method, and for producing the shape, it uses an interpolation scheme. The results show that the distinction between shape and texture in the algorithm improves the results both in detecting the rainfall and in producing the accurate rain intensity.

**[2] ÖZMEN, Ramazan, and Mustafa Günay. "An Investigation on Deformation Behaviors of Energy Absorbers for Passenger Coaches." Periodicals of Engineering and Natural Sciences (PEN) 5, no. 3 (2017).**

Thin-walled structures is used commonly as energy absorbers at the front and back of the coaches. These parts should be designed to minimize the damage to the vehicle and prevent the passengers from fatality and/or injury by absorbing the collision energy in railway transportation. In this paper, deformation behaviors of tube like structures with truncated cone under the axial impact load were investigated by means of finite element analysis (FEA). The energy absorbers having tube like structures were modelled at the same weight and have three different wall thickness and taper angle. As a result of FEA, the performances of straight and truncated cone type energy absorbers were compared in terms of energy absorption capacities and an optimization study was done to determine the effects of thickness and taper angle on energy absorbing performances of the members. The analysis of variance in 95% confidence level was applied in order to determine the effects of design parameters on total efficiency (TE). Besides, optimum design parameters for TE were determined by using Taguchi optimization methodology. Thickness was found as the most significant parameter on total efficiency with 60.52% percentage contribution ratio according to ANOVA results.

**[3] Sharif, Md Haidar, Ivan Despot, and Sahin Uyaver. "A proof of concept for home automation system with implementation of the internet of things standards." Periodicals of Engineering and Natural Sciences (PEN) 6, no. 1 (2018): 95-106.**

A key issue in soil moisture data assimilation is that observational and modeling uncertainties are poorly known, and incorrect assumptions about these errors may compromise the efficiency of the land data assimilation system. It is thus crucial to investigate the impact of the error characterization on the assimilation of soil moisture observations, in particular because LDASs often use very simplistic error models. As rainfall is the dominant meteorological forcing input to the land surface model for soil moisture estimation, a more comprehensive characterization of rainfall uncertainty may improve soil moisture estimates. Since soil moisture temporally integrates antecedent precipitation and is subject to lower and upper limits, the variability of errors in soil moisture is typically smaller than that of errors in precipitation. This error variance relationship is not linear and depends on the error properties of the rainfall fields showed that the use of a complex error model to characterize the spatial variability of rainfall errors could better capture soil moisture error properties. Furthermore, in a synthetic numerical assimilation experiment demonstrated that using the more elaborate rainfall error model may slightly improve surface and root zone soil moisture estimates obtained from assimilating soil moisture retrievals. Specifically, five experiments were conducted: a “benchmark” simulation forced with Stage IV radar rainfall, and four experiments obtained by perturbing satellite rainfall fields with the two rainfall error models of different complexity, with and without the assimilation of AMSR-E soil moisture retrievals. Satellite rainfall was from the NOAA Climate Prediction Center morphing product. Surface and root zone soil moisture outputs from each experiment were compared against Oklahoma Mesonet in situ measurements.

**[4] Fraiwan, L., et al. A wireless home safety gas leakage detection system. in Biomedical Engineering (MECBME), 2011 1st Middle East Conference on. 2011. IEEE.**

The National Mosaic and Multi-Sensor Quantitative Precipitation Estimation system is a multiradar, multisensor system built upon the CRAFT data network. The objectives of NMQ research and development are to assimilate different observational networks toward creating 1) high-resolution national multisensor QPEs for flash flood and flood warnings and water resource management and 2) high-resolution national 3D grids of radar reflectivity (Z) for data assimilation, numerical weather prediction model verification, and aviation product development. The system ingests base-level data from more than 140 WSR-88D radars and about 31 Canadian C-band weather radars and generates 3D radar reflectivity mosaic and QPE products in real time over the conterminous United States. In addition, the NMQ system ingests Rapid Update Cycle model analysis fields and Hydro meteorological Automated Data System gauge data for various multisensor QPE algorithms. Further, the NMQ system has a Web-based evaluation tool that ingests a number of operational QPE products generated from different sensors where the NMQ experimental products are compared with the operational products and with independent gauge observations in real time. The Web-based verification system is accessible to the NWS forecasters and to the general public. Feedbacks from the forecasters and public provided valuable guidance for the R&D of NMQ.

**[5] Mustafa, K. and H. Gitano-Briggs. Liquefied petroleum gas (LPG) as an alternative fuel in spark ignition engine: Performance and emission characteristics. in Energy and Environment, 2009. ICEE 2009. 3rd International Conference on. 2009. IEEE.**

In the present study, the performance and emission characteristics of a four-stroke spark ignition engine operated on liquefied petroleum gas (LPG) were investigated experimentally. The LPG was supplied from a LPG tank which was purchased from a local gas distributor. The primary content of LPG is 60% propane and 40% butane. The four-stroke spark ignition engine has an engine capacity of 183 cc and a compression ratio of 6.3:1, and it was coupled to a 5 kW eddy current dynamometer for performance measurement. A 5-gas Non-Dispersive Infra-Red (NDIR) analyzer was used for CO, CO 2 , unburnt HC, and NO x measurement. Two sets of experimental data obtained were analyzed; (i) 100% gasoline, and (ii) 5%, 10%, and 20% of LPG in gasoline. It was found that in general, the engine's power output and torque suffer a drop in performance compared to 100% gasoline fueled engine, when tests were evaluated with 5%, 10%, and 20% LPG in gasoline. However, the brake specific fuel consumption, BSFC shows an improvement with LPG as a fuel replacement. The concentration levels of CO, CO 2 , unburnt HC, and NO x recorded are found to be lower than the gasoline fueled engine.

**CHAPTER-3**

**EXISTING METHOD**

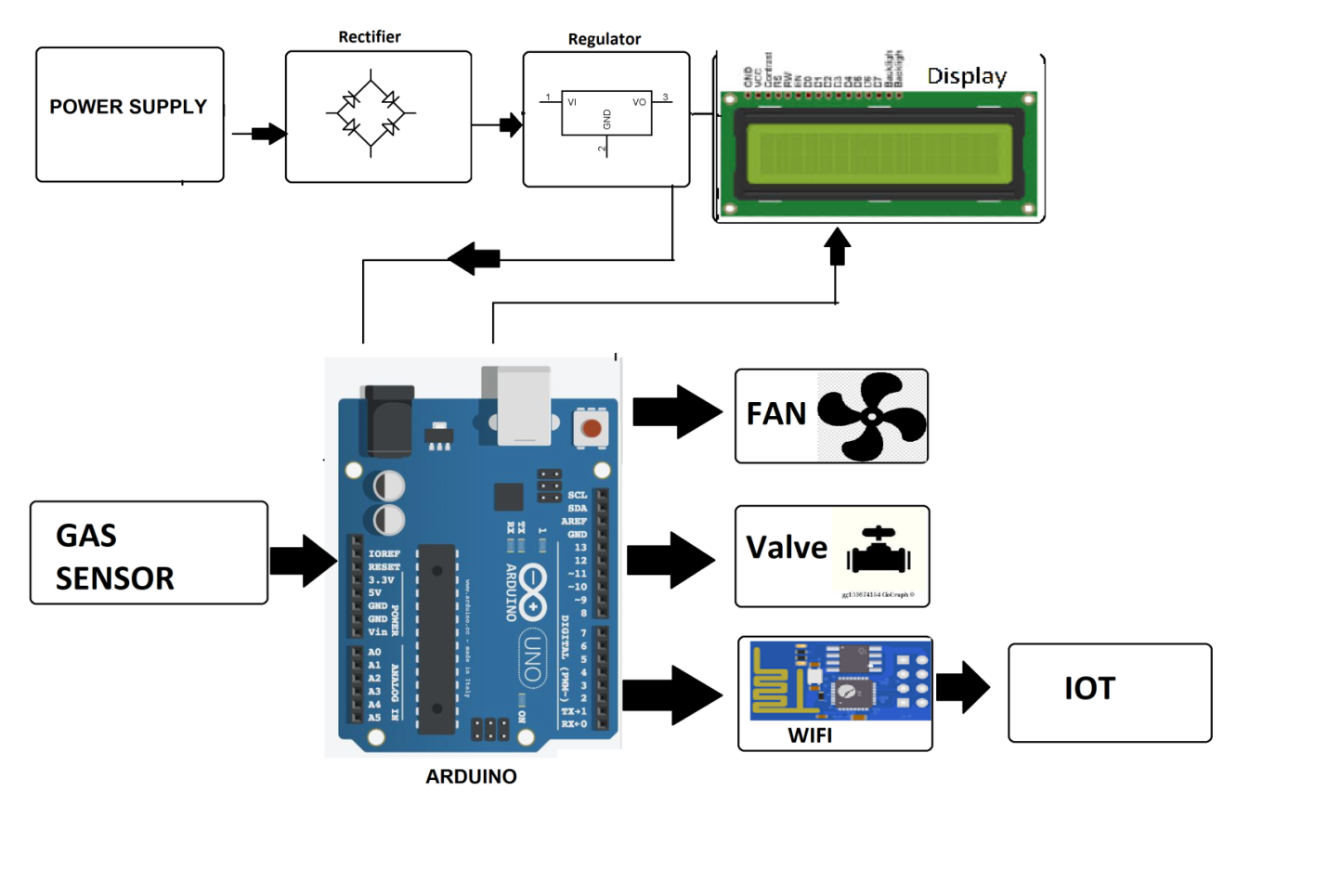
In existing system most of LPG explosions are caused by undetected gas leakage in the pre-detection condition. So that, LPG detection system is needed. The purpose of this system is to detect gas leakage, neutralize it, and prevent the explosion. Gas leakage could happen due to improper regulator installation or the hose is broken. This detection should not work in just one location because gas can leak at the gas regulator and its hose. Therefore, Wireless Sensor Network (WSN) is one of the methods that suitable for detecting gas leakage in the wider area. This method uses two or more gas sensors to detect leakage in two or more locations around the gas tube and its distribution line. WSN system works based on gas sensor MQ-6 and wireless module Bluetooth HC-05. Explosion prevention system works based on alarm/buzzer, exhaust fan, and automatic gas regulator. If the gas leaks, the sensor will send its data wirelessly to Arduino. Then, explosion prevention system will be activated. The system will turn the alarm/buzzer on, automatically releases gas regulator, and neutralizes the air with the exhaust fan. Both systems will be fully controlled by Arduino platform.

**CHAPTER-4**

**PROPOSED SYSTEM**

In proposed system gas leakage is becoming a major concern at all the places wherever it is being used as it may lead to a small accident or sometimes major disaster causing a great loss of infrastructures and lives of people. With the available advanced IoT and Mobile technologies, it is possible to build a system to prevent such accidents and informing the right person in case of tragedy. Hence, this paper has designed and developed a device which is capable of detecting the gas leakage and send alert message to specified mobile number. This proposed device could detect leakage of LPG, Methane, Butane or any such petroleum based gaseous substance using MQ5 Sensor (gas sensor) and send SMS to registered mobile number (by inserting sim in GSM module) and when the gas leakage is detected by the gas sensor then it sends an SMS alert to the registered mobile number thus notifying the owner to be careful and to take careful measures thus saving their lives and things. Implementation results of this device have proved its effectiveness

**BLOCK DIAGRAM**

****

**ADVANTAGE**

* This project has applications in our home. We can also use this project in industries and offices and colleges where the LPG gas cylinder is used in the canteen.
* This project also has applications in hotels and restaurants.
* The main Advantage of the Arduino Uno based LPG detector system project is that it gives a remote indication to the user about the LPG leakage with the help of SMS indication.
* To enhance this project, we can add a GPS modem to this system.

**APPLICATION**

* Roadside pollution Monitoring.
* Industrial Perimeter Monitoring.
* Site selection for reference monitoring stations.
* Indoor Air Quality Monitoring.
* To make this data available to the common man.

**CHAPTER 5**

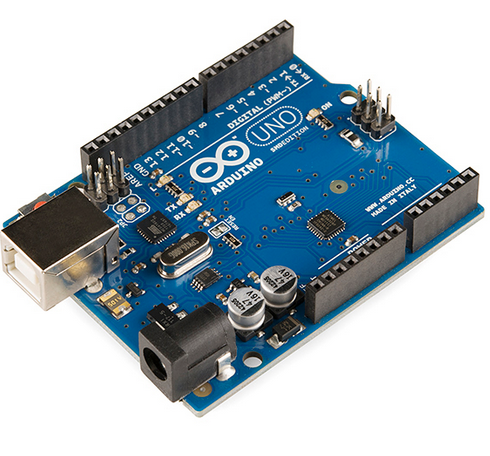
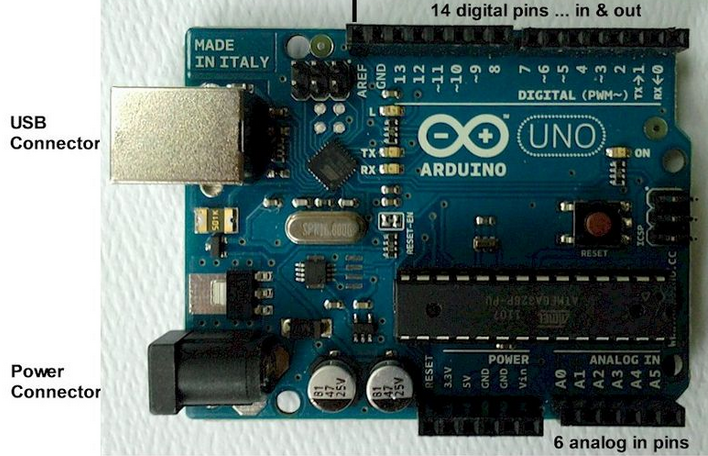
**HARDWARE DESCRIPTION**

**ARDUINO**

**Arduino** is a computer hardware and software company, project, and user community that designs and manufactures [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as [open-source hardware](https://en.wikipedia.org/wiki/Open-source_hardware) and [software](https://en.wikipedia.org/wiki/Open-source_software), which are licensed under the [GNU Lesser General Public License](https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License) (LGPL) or the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL),[[1]](https://en.wikipedia.org/wiki/Arduino#cite_note-1) permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as [do-it-yourself](https://en.wikipedia.org/wiki/Do-it-yourself) kits.

The project's board designs use a variety of microprocessors and controllers. These systems provide sets of digital and analog [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O) pins that may be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus ([USB](https://en.wikipedia.org/wiki/USB)) on some models, for loading programs from personal computers. The microcontrollers are mainly programmed using a dialect of features from the programming languages [C](https://en.wikipedia.org/wiki/C_%28programming_language%29) and [C++](https://en.wikipedia.org/wiki/C%2B%2B). In addition to using traditional compiler toolchains, the Arduino project provides an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) based on the [Processing](https://en.wikipedia.org/wiki/Processing_%28programming_language%29) language project.

The Arduino project started in 2005 as a program for students at the [Interaction Design Institute Ivrea](https://en.wikipedia.org/wiki/Interaction_Design_Institute_Ivrea) in [Ivrea](https://en.wikipedia.org/wiki/Ivrea), Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using [sensors](https://en.wikipedia.org/wiki/Sensor) and [actuators](https://en.wikipedia.org/wiki/Actuator). Common examples of such devices intended for beginner hobbyists include simple [robots](https://en.wikipedia.org/wiki/Robot), [thermostats](https://en.wikipedia.org/wiki/Thermostat), and [motion detectors](https://en.wikipedia.org/wiki/Motion_detector)

****

**Daigram of Arduino Uno SMD R3**

|  |  |
| --- | --- |
| Developer | Arduino |
| Manufacturer | Many |
| Type | [Single-board microcontroller](https://en.wikipedia.org/wiki/Single-board_microcontroller) |
| [Operating system](https://en.wikipedia.org/wiki/Operating_system) | None |
| [CPU](https://en.wikipedia.org/wiki/Central_processing_unit) | [Atmel AVR](https://en.wikipedia.org/wiki/Atmel_AVR) (8-bit), [ARM Cortex-M0+](https://en.wikipedia.org/wiki/ARM_Cortex-M0%2B) (32-bit), [ARM Cortex-M3](https://en.wikipedia.org/wiki/ARM_Cortex-M3) (32-bit), [Intel Quark](https://en.wikipedia.org/wiki/Intel_Quark) ([x86](https://en.wikipedia.org/wiki/X86)) (32-bit) |
| Memory | [SRAM](https://en.wikipedia.org/wiki/Static_random-access_memory) |
| Storage | [Flash](https://en.wikipedia.org/wiki/Flash_memory), [EEPROM](https://en.wikipedia.org/wiki/EEPROM) |

**AVR**

The **AVR** is a [modified Harvard architecture](http://en.wikipedia.org/wiki/Modified_Harvard_architecture) [8-bit](http://en.wikipedia.org/wiki/8-bit) [RISC](http://en.wikipedia.org/wiki/Reduced_instruction_set_computer) single-chip [microcontroller](http://en.wikipedia.org/wiki/Microcontroller), which was developed by [Atmel](http://en.wikipedia.org/wiki/Atmel) in 1996. The AVR was one of the first microcontroller families to use on-chip [flash memory](http://en.wikipedia.org/wiki/Flash_memory) for program storage, as opposed to [one-time programmable ROM](http://en.wikipedia.org/wiki/Programmable_read-only_memory),[EPROM](http://en.wikipedia.org/wiki/EPROM), or [EEPROM](http://en.wikipedia.org/wiki/EEPROM) used by other microcontrollers at the time.

The AVR is a [modified Harvard architecture](http://en.wikipedia.org/wiki/Modified_Harvard_architecture) machine, where program and data are stored in separate physical memory systems that appear in different address spaces, but having the ability to read data items from program memory using special instructions.

**Basic families**

AVRs are generally classified into following:

**tinyAVR** — the [ATtiny series](http://en.wikipedia.org/wiki/Atmel_AVR_ATtiny_comparison_chart" \o "Atmel AVR ATtiny comparison chart)

* + 0.5–16 kB program memory
  + 6–32-pin package
  + Limited peripheral set

**megaAVR** — the ATmega series

* + 4–512 kB program memory
  + 28–100-pin package
  + Extended instruction set (multiply instructions and instructions for handling larger program memories)
  + Extensive peripheral set

**XMEGA** — the ATxmega series

* + 16–384 kB program memory
  + 44–64–100-pin package (A4, A3, A1)
  + Extended performance features, such as DMA, "Event System", and cryptography support.
  + Extensive peripheral set with [ADCs](http://en.wikipedia.org/wiki/Analog-to-digital_converter)

**Application-specific AVR**

* + megaAVRs with special features not found on the other members of the AVR family, such as LCD controller, [USB](http://en.wikipedia.org/wiki/Universal_Serial_Bus) controller, advanced PWM, CAN, etc.

**FPSLIC (AVR with FPGA)**

* + [FPGA](http://en.wikipedia.org/wiki/Field-programmable_gate_array) 5K to 40K gates
  + SRAM for the AVR program code, unlike all other AVRs
  + AVR core can run at up to 50 MHZ

**32-bit AVRs**

In 2006 Atmel released microcontrollers based on the 32-bit [AVR32](http://en.wikipedia.org/wiki/AVR32) architecture. They include [SIMD](http://en.wikipedia.org/wiki/SIMD) and [DSP](http://en.wikipedia.org/wiki/Digital_signal_processor) instructions, along with other audio- and video-processing features. This 32-bit family of devices is intended to compete with the [ARM](http://en.wikipedia.org/wiki/ARM_architecture)-based processors. The instruction set is similar to other RISC cores, but it is not compatible with the original AVR or any of the various ARM cores.

**Device architecture**

[Flash](http://en.wikipedia.org/wiki/Flash_memory), [EEPROM](http://en.wikipedia.org/wiki/EEPROM), and [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory) are all integrated onto a single chip, removing the need for external memory in most applications. Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices. Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

**Program memory**

Program instructions are stored in [non-volatile](http://en.wikipedia.org/wiki/Non-volatile) [flash memory](http://en.wikipedia.org/wiki/Flash_memory). Although the [MCUs](http://en.wikipedia.org/wiki/Microcontroller_unit) are 8-bit, each instruction takes one or two 16-bit words.

The size of the program memory is usually indicated in the naming of the device itself (e.g., the ATmega64x line has 64 kB of flash, while the ATmega32x line has 32 kB).

There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash. However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

**Internal data memory**

The data [address space](http://en.wikipedia.org/wiki/Address_space) consists of the [register file](http://en.wikipedia.org/wiki/Register_file), I/O registers, and [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory).

**Internal registers**

The AVRs have 32 [single-byte](http://en.wikipedia.org/wiki/Byte) [registers](http://en.wikipedia.org/wiki/Processor_register) and are classified as 8-bit RISC devices.

In the tinyAVR and megaAVR variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses (000016–001F16), followed by 64 I/O registers (002016–005F16). In devices with many peripherals, these registers are followed by 160 “extended I/O” registers, only accessible as [memory-mapped I/O](http://en.wikipedia.org/wiki/Memory-mapped_I/O) (006016–00FF16).

Actual SRAM starts after these register sections, at address 006016 or, in devices with “extended I/O”, at 010016.

Even though there are separate addressing schemes and optimized opcodes for accessing the register file and the first 64 I/O registers, all can still be addressed and manipulated as if they were in SRAM.

The very smallest of the tinyAVR variants use a reduced architecture with only 16 registers (r0 through r15 are omitted) which are not addressable as memory locations. I/O memory begins at address 000016, followed by SRAM. In addition, these devices have slight deviations from the standard AVR instruction set. Most notably, the direct load/store instructions (LDS/STS) have been reduced from 2 words (32 bits) to 1 word (16 bits), limiting the total direct addressable memory (the sum of both I/O and SRAM) to 128 bytes. Conversely, the indirect load instruction's (LD) 16-bit address space is expanded to also include non-volatile memory such as Flash and configuration bits; therefore, the LPM instruction is unnecessary and omitted.

In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM. Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space. Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes (000016–0FFF16). As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions). Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space, which can be used optionally for mapping the internal EEPROM to the data address space (100016–1FFF16). The actual SRAM is located after these ranges, starting at 200016.

**GPIO ports**

Each [GPIO](http://en.wikipedia.org/wiki/GPIO) port on a tiny or mega AVR drives up to eight pins and is controlled by three 8-bit registers: DDRx, PORTx and PINx, where x is the port identifier.

* DDRx: Data Direction Register, configures the pins as either inputs or outputs.
* PORTx: Output port register. Sets the output value on pins configured as outputs. Enables or disables the [pull-up resistor](http://en.wikipedia.org/wiki/Pull-up_resistor) on pins configured as inputs.
* PINx: Input register, used to read an input signal. On some devices (but not all, check the datasheet), this register can be used for pin toggling: writing a logic one to a PINx bit toggles the corresponding bit in PORTx, irrespective of the setting of the DDRx biT

xmegaAVR have additional registers for push/pull, totem-pole and pullup configurations.

**EEPROM**

Almost all AVR microcontrollers have internal [EEPROM](http://en.wikipedia.org/wiki/EEPROM) for semi-permanent data storage. Like flash memory, EEPROM can maintain its contents when electrical power is removed.

In most variants of the AVR architecture, this internal EEPROM memory is not mapped into the MCU's addressable memory space. It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions, which makes EEPROM access much slower than other internal RAM.

However, some devices in the SecureAVR (AT90SC) family use a special EEPROM mapping to the data or program memory, depending on the configuration. The XMEGA family also allows the EEPROM to be mapped into the data address space.

Since the number of writes to EEPROM is not unlimited — Atmel specifies 100,000 write cycles in their datasheets — a well designed EEPROM write routine should compare the contents of an EEPROM address with desired contents and only perform an actual write if the contents need to be changed.

Note that erase and write can be performed separately in many cases, byte-by-byte, which may also help prolong life when bits only need to be set to all 1s (erase) or selectively cleared to 0s (write).

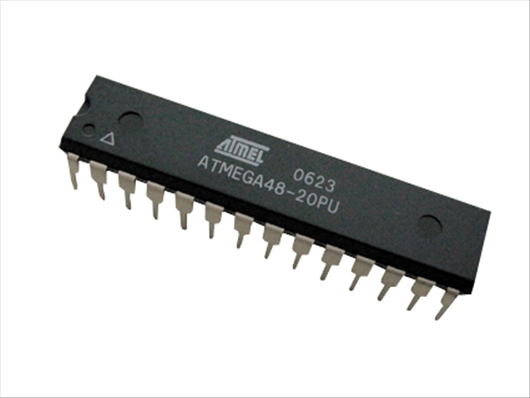


FIG NO: ATMEGA8A

**Program execution**

Atmel's AVRs have a two-stage, single-level [pipeline](http://en.wikipedia.org/wiki/Pipeline_(computing)) design. This means the next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVRs relatively fast among [eight-bit](http://en.wikipedia.org/wiki/Eight-bit) microcontrollers.

The AVR processors were designed with the efficient execution of [compiled](http://en.wikipedia.org/wiki/Compiler) [C](http://en.wikipedia.org/wiki/C_(programming_language)) code in mind and have several built-in pointers for the task.

**Instruction set**

The [AVR instruction set](http://en.wikipedia.org/wiki/Atmel_AVR_instruction_set) is more [orthogonal](http://en.wikipedia.org/wiki/Orthogonal_instruction_set) than those of most eight-bit microcontrollers, in particular the [8051 clones](http://en.wikipedia.org/wiki/Intel_8051) and [PIC microcontrollers](http://en.wikipedia.org/wiki/PIC_microcontroller) with which AVR competes today. However, it is not completely regular:

* [Pointer registers](http://en.wikipedia.org/wiki/Pointer_register) X, Y, and Z have addressing capabilities that are different from each other.
* [Register](http://en.wikipedia.org/wiki/Processor_register) locations R0 to R15 have different addressing capabilities than register locations R16 to R31.
* I/O ports 0 to 31 have different addressing capabilities than I/O ports 32 to 63.
* CLR affects flags, while SER does not, even though they are complementary instructions. CLR set all bits to zero, and SER sets them to one. (Note that CLR is pseudo-op for EOR R, R; and SER is short for LDI R,$FF. Math operations such as EOR modify flags, while moves/loads/stores/branches such as LDI do not.)
* Accessing read-only data stored in the program memory (flash) requires special LPM instructions; the flash bus is otherwise reserved for instruction memory.

Additionally, some chip-specific differences affect code generation. Code pointers (including return addresses on the stack) are two bytes long on chips with up to 128 kBytes of flash memory, but three bytes long on larger chips; not all chips have hardware multipliers; chips with over 8 kB of flash have branch and call instructions with longer ranges; and so forth.

The mostly regular instruction set makes programming it using C (or even Ada) compilers fairly straightforward. [GCC](http://en.wikipedia.org/wiki/GNU_Compiler_Collection) has included AVR support for quite some time, and that support is widely used. In fact, Atmel solicited input from major developers of compilers for small microcontrollers, to determine the instruction set features that were most useful in a compiler for high-level languages.

**MCU speed**

The AVR line can normally support clock speeds from 0 to 20 MHz, with some devices reaching 32 MHz. Lower-powered operation usually requires a reduced clock speed. All recent (Tiny, Mega, and Xmega, but not 90S) AVRs feature an on-chip oscillator, removing the need for external clocks or resonator circuitry. Some AVRs also have a system clock prescaler that can divide down the system clock by up to 1024. This prescaler can be reconfigured by software during run-time, allowing the clock speed to be optimized.

Since all operations (excluding multiplication and 16-bit add/subtract) on registers R0–R31 are single-cycle, the AVR can achieve up to 1 [MIPS](http://en.wikipedia.org/wiki/Million_instructions_per_second) per MHz, i.e. an 8 MHz processor can achieve up to 8 MIPS. Loads and stores to/from memory take two cycles, branching takes two cycles. Branches in the latest "3-byte PC" parts such as ATmega2560 are one cycle slower than on previous devices.

**Development**

AVRs have a large following due to the free and inexpensive development tools available, including reasonably priced development boards and free development software. The AVRs are sold under various names that share the same basic core, but with different peripheral and memory combinations. Compatibility between chips in each family is fairly good, although I/O controller features may vary.

**Features**

AVRs offer a wide range of features:

* Multifunction, bi-directional general-purpose I/O ports with configurable, built-in [pull-up resistors](http://en.wikipedia.org/wiki/Pull-up_resistor)
* Multiple internal oscillators, including RC oscillator without external parts
* Internal, self-programmable instruction [flash memory](http://en.wikipedia.org/wiki/Flash_memory) up to 256 kB (384 kB on XMega)
  + [In-system programmable](http://en.wikipedia.org/wiki/In-system_programming) using serial/parallel low-voltage proprietary interfaces or [JTAG](http://en.wikipedia.org/wiki/JTAG)
  + Optional boot code section with independent lock bits for protection
* On-chip debugging (OCD) support through JTAG or [debugWIRE](http://en.wikipedia.org/wiki/DebugWIRE" \o "DebugWIRE) on most devices
  + The JTAG signals (TMS, TDI, TDO, and TCK) are multiplexed on [GPIOs](http://en.wikipedia.org/wiki/General_Purpose_Input/Output). These pins can be configured to function as JTAG or GPIO depending on the setting of a fuse bit, which can be programmed via ISP or HVSP. By default, AVRs with JTAG come with the JTAG interface enabled.
  + [debug WIRE](http://en.wikipedia.org/wiki/DebugWIRE) uses the /RESET pin as a bi-directional communication channel to access on-chip debug circuitry. It is present on devices with lower pin counts, as it only requires one pin.
* Internal data [EEPROM](http://en.wikipedia.org/wiki/EEPROM) up to 4 kB
* Internal [SRAM](http://en.wikipedia.org/wiki/Static_random-access_memory) up to 16 kB (32 kB on XMega)
* External 64 kB little endian data space on certain models, including the Mega8515 and Mega162.
  + The external data space is overlaid with the internal data space, such that the full 64 kB address space does not appear on the external bus and accesses to e.g. address 010016 will access internal RAM, not the external bus.
  + In certain members of the XMega series, the external data space has been enhanced to support both SRAM and SDRAM. As well, the data addressing modes have been expanded to allow up to 16 MB of data memory to be directly addressed.
  + AVRs generally do not support executing code from external memory. Some [ASSPs](http://en.wikipedia.org/wiki/Application-specific_standard_product) using the AVR core do support external program memory.
* 8-bit and 16-bit timers
  + [PWM](http://en.wikipedia.org/wiki/Pulse-width_modulation) output (some devices have an enhanced PWM peripheral which includes a dead-time generator)
  + [Input capture](http://en.wikipedia.org/wiki/Input_capture) that record a time stamp triggered by a signal edge
* Analog comparator
* 10 or 12-bit [A/D converters](http://en.wikipedia.org/wiki/Analog-to-digital_converter), with multiplex of up to 16 channels
* 12-bit [D/A converters](http://en.wikipedia.org/wiki/Digital-to-analog_converter)
* A variety of serial interfaces, including
  + [I²C](http://en.wikipedia.org/wiki/I%C2%B2C) compatible Two-Wire Interface (TWI)
  + Synchronous/asynchronous serial peripherals ([UART](http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter)/USART) (used with [RS-232](http://en.wikipedia.org/wiki/RS-232), [RS-485](http://en.wikipedia.org/wiki/RS-485), and more)
  + [Serial Peripheral Interface Bus](http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus) (SPI)
  + Universal Serial Interface (USI): a multi-purpose hardware communication module that can be used to implement an SPI I2C or UART interface.
* [Brownout](http://en.wikipedia.org/wiki/Brownout_(electricity)) detection
* [Watchdog timer](http://en.wikipedia.org/wiki/Watchdog_timer) (WDT)
* Multiple power-saving sleep modes
* Lighting and motor control ([PWM](http://en.wikipedia.org/wiki/Pulse-width_modulation)-specific) controller models
* [CAN](http://en.wikipedia.org/wiki/Controller_area_network) controller support
* [USB](http://en.wikipedia.org/wiki/Universal_Serial_Bus) controller support
  + Proper full-speed (12 Mbit/s) hardware & Hub controller with embedded AVR.
  + Also freely available low-speed (1.5 Mbit/s) ([HID](http://en.wikipedia.org/wiki/Human_interface_device)) [bit banging](http://en.wikipedia.org/wiki/Bit-banging) software emulations
* [Ethernet](http://en.wikipedia.org/wiki/Ethernet) controller support
* [LCD](http://en.wikipedia.org/wiki/Liquid_crystal_display) controller support
* Low-voltage devices operating down to 1.8 V (to 0.7 V for parts with built-in DC–DC up converter)
* picoPower devices
* [DMA](http://en.wikipedia.org/wiki/Direct_memory_access) controllers and "event system" peripheral communication.
* Fast cryptography support for [AES](http://en.wikipedia.org/wiki/Advanced_Encryption_Standard) and [DES](http://en.wikipedia.org/wiki/Data_Encryption_Standard)

**LCD:**

A liquid crystal display (LCD) is a [flat panel display](http://en.wikipedia.org/wiki/Flat_panel_display), [electronic visual display](http://en.wikipedia.org/wiki/Electronic_visual_display), or [video display](http://en.wikipedia.org/wiki/Video_display) that uses the light modulating properties of [liquid crystals](http://en.wikipedia.org/wiki/Liquid_Crystals). Liquid crystals do not emit light directly. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images which can be displayed or hidden, such as preset words, digits, and [7-segment](http://en.wikipedia.org/wiki/7-segment) displays as in a [digital clock](http://en.wikipedia.org/wiki/Digital_clock). They use the same basic technology, except that arbitrary images are made up of a large number of small [pixels](http://en.wikipedia.org/wiki/Pixel), while other displays have larger elements. LCDs are used in a wide range of applications including [computer monitors](http://en.wikipedia.org/wiki/Computer_monitor), [televisions](http://en.wikipedia.org/wiki/Television), [instrument panels](http://en.wikipedia.org/wiki/Instrument_panel), [aircraft cockpit displays](http://en.wikipedia.org/wiki/Flight_instruments), and signage. They are common in consumer devices such as video players, gaming devices, [clocks](http://en.wikipedia.org/wiki/Clock), [watches](http://en.wikipedia.org/wiki/Watch), [calculators](http://en.wikipedia.org/wiki/Calculator), and [telephones](http://en.wikipedia.org/wiki/Telephone), and have replaced [cathode ray tube](http://en.wikipedia.org/wiki/Cathode_ray_tube) (CRT) displays in most applications. They are available in a wider range of screen sizes than CRT and [plasma displays](http://en.wikipedia.org/wiki/Plasma_display), and since they do not use phosphors, they do not suffer [image burn-in](http://en.wikipedia.org/wiki/Screen_burn-in). LCDs are, however, susceptible to [image persistence](http://en.wikipedia.org/wiki/Image_persistence).

The LCD screen is more energy efficient and can be disposed of more safely than a CRT. Its low electrical power consumption enables it to be used in [battery](http://en.wikipedia.org/wiki/Battery_(electricity))-powered [electronic](http://en.wikipedia.org/wiki/Electronics) equipment. It is an [electronically modulated optical device](http://en.wikipedia.org/wiki/Electro-optic_modulator) made up of any number of segments filled with [liquid crystals](http://en.wikipedia.org/wiki/Liquid_crystal) and arrayed in front of a [light source](http://en.wikipedia.org/wiki/Light#Light_sources) ([backlight](http://en.wikipedia.org/wiki/Backlight)) or [reflector](http://en.wikipedia.org/wiki/Reflector_(photography)) to produce images in color or [monochrome](http://en.wikipedia.org/wiki/Monochrome). Liquid crystals were first discovered in 1888. By 2008, worldwide sales of televisions with LCD screens exceeded annual sales of CRT units; the CRT became obsolete for most purposes.



Buzzer



An **arduino buzzer** is also called a piezo **buzzer**. It is basically a tiny speaker that you can connect directly to an **Arduino**. You can make it sound a tone at a frequency you set. The **buzzer** produces sound based on reverse of the piezoelectric effect.

Relay

A **relay** is an [electrically](https://en.wikipedia.org/wiki/Electric) operated [switch](https://en.wikipedia.org/wiki/Switch). It consists of a set of input terminals for a single or multiple control signals, and a set of operating contact terminals. The switch may have any number of contacts in multiple [contact forms](https://en.wikipedia.org/wiki/Electrical_contact#Contact_form), such as make contacts, break contacts, or combinations thereof.

Relays are used where it is necessary to control a circuit by an independent low-power signal, or where several circuits must be controlled by one signal. Relays were first used in long-distance [telegraph](https://en.wikipedia.org/wiki/Electrical_telegraph) circuits as signal repeaters: they refresh the signal coming in from one circuit by transmitting it on another circuit. Relays were used extensively in telephone exchanges and early computers to perform logical operations.



**SOFTWARE DESCRIPTION**

**Embedded C**

An embedded system is an application that contains at least one programmable computer (typically in the form of a microcontroller, a microprocessor or digital signal processor chip) and which is used by individuals who are, in the main, unaware that the system is computer-based.

**Introduction**

Looking around, we find ourselves to be surrounded by various types of [embedded systems](http://www.engineersgarage.com/articles/embedded-systems). Be it a digital camera or a mobile phone or a washing machine, all of them has some kind of processor functioning inside it. Associated with each processor is the embedded software. If hardware forms the body of an embedded system, embedded processor acts as the brain, and embedded software forms its soul. It is the embedded software which primarily governs the functioning of embedded systems.

During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check correct execution of the program. Some ‘very fortunate’ developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well.

As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

Initially C was developed by Kernighan and Ritchie to fit into the space of 8K and to write (portable) operating systems. Originally it was implemented on UNIX operating systems. As it was intended for operating systems development, it can manipulate memory addresses. Also, it allowed programmers to write very compact codes. This has given it the reputation as the language of choice for hackers too.

As assembly language programs are specific to a processor, assembly language didn’t offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn’t find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C. Subsequent sections will discuss what is Embedded C, features of C language, similarities and difference between C and embedded C, and features of embedded C programming.

**EMBEDDED SYSTEMS PROGRAMMING**

Embedded systems programming is different from developing applications on a desktop computers. Key characteristics of an embedded system, when compared to PCs, are as follows. Embedded devices have resource constraints(limited ROM, limited RAM, limited stack space, less processing power) Components used in embedded system and PCs are different; embedded systems typically uses smaller, less power consuming components.  Embedded systems are more tied to the hardware.

Two salient features of Embedded Programming are code speed and code size. Code speed is governed by the processing power, timing constraints, whereas code size is governed by available program memory and use of programming language.  Goal of embedded system programming is to get maximum features in minimum space and minimum time.

Embedded systems are programmed using different type of language

* Machine Code
* Low level language, i.e., assembly
* High level language like C, C++, Java, Ada, etc.
* Application level language like Visual Basic, scripts, Access, etc.

Assembly language maps mnemonic words with the binary machine codes that the processor uses to code the instructions. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high level languages are preferred for embedded systems programming.

Use of C in embedded systems is driven by following advantages it is small and reasonably simpler to learn, understand, program and debug. C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.

·

 Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems. As C combines functionality of assembly language and features of high level languages, C is treated as a ‘middle-level computer language’ or ‘high level assembly language’. It is fairly efficient. It supports access to I/O and provides ease of management of large embedded projects.

Many of these advantages are offered by other languages also, but what sets C apart from others like Pascal, FORTRAN, etc. is the fact that it is a middle level language; it provides direct hardware control without sacrificing benefits of high level languages. Compared to other high level languages, C offers more flexibility because C is relatively small, structured language; it supports low-level bit-wise data manipulation.

Compared to assembly language, C Code written is more reliable and scalable, more portable between different platforms (with some changes). Moreover, programs developed in C are much easier to understand, maintain and debug. Also, as they can be developed more quickly, codes written in C offers better productivity. C is based on the philosophy ‘programmers know what they are doing’; only the intentions are to be stated explicitly. It is easier to write good code in C & convert it to an efficient assembly code (using high quality compilers) rather than writing an efficient code in assembly itself. Benefits of assembly language programming over C are negligible when we compare the ease with which C programs are developed by programmers.

Objected oriented language, C++ is not apt for developing efficient programs in resource constrained environments like embedded devices. Virtual functions & exception handling of C++ are some specific features that are not efficient in terms of space and speed in embedded systems. Sometimes C++ is used only with very few features, very much as C.

Ada, also an object-oriented language, is different than C++. Originally designed by the U.S. DOD, it didn’t gain popularity despite being accepted as an international standard twice (Ada83 and Ada95). However, Ada language has many features that would simplify embedded software development.

Java is another language used for embedded systems programming. It primarily finds usage in high-end mobile phones as it offers portability across systems and is also useful for browsing applications. Java programs require Java Virtual Machine (JVM), which consume lot of resources. Hence it is not used for smaller embedded devices. Dynamic C and B# are some proprietary languages which are also being used in embedded applications. Efficient embedded [C programs](http://www.engineersgarage.com/c-language-programs) must be kept small and efficient; they must be optimized for code speed and code size. Good understanding of processor architecture embedded C programming and debugging tools facilitate this.

 Difference between C and embedded C:

Though C and embedded C appear different and are used in different contexts, they have more similarities than the differences. Most of the constructs are same; the difference lies in their applications.

C is used for desktop computers, while embedded C is for microcontroller based applications. Accordingly, C has the luxury to use resources of a desktop PC like memory, OS, etc. While programming on desktop systems, we need not bother about memory. However, embedded C has to use with the limited resources (RAM, ROM, I/Os) on an embedded processor. Thus, program code must fit into the available program memory. If code exceeds the limit, the system is likely to crash.

Compilers for C (ANSI C) typically generate OS dependant executables. Embedded C requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run. Embedded compilers give access to all resources which is not provided in compilers for desktop computer applications. Embedded systems often have the real-time constraints, which is usually not there with desktop computer applications.

Embedded systems often do not have a console, which is available in case of desktop applications. So, what basically is different while programming with embedded C is the mindset; for embedded applications, we need to optimally use the resources, make the program code efficient, and satisfy real time constraints, if any. All this is done using the basic constructs, syntaxes, and function libraries of ‘C’.

**CODING**

#include <LiquidCrystal.h>

const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;  
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);  
const int s1 =  A0;      
  
  
int Ss1 = 0;  
  
  
int i=0,j=0;  
             
  
void setup()  
{  
Serial.begin(9600);  
pinMode(2,OUTPUT);  
pinMode(3,OUTPUT);  
  
digitalWrite(2,HIGH);  
digitalWrite(3,LOW);  
  
 delay(100);  
  lcd.begin(16, 2);  
lcd.print("GAS LEAKAGE DETECT");  
lcd.setCursor(0,1) ;  
lcd.print("    SYSTEM    ");  
  
}  
  
void loop()  
{  
Ss1 = analogRead(s1);  
lcd.begin(16, 2);  
lcd.setCursor(0,0) ;  
lcd.print("MONITOR--->>>>>");  
lcd.setCursor(0,1) ;  
lcd.print("GAS LAVEL:");  
lcd.print(Ss1);  
delay(100);  
  
  
  
  
  
  
  
   
    if(Ss1>=700)  
    {  
   
    lcd.setCursor(3,1) ;  
    lcd.print("YES");  
    lcd.setCursor(0,0) ;  
    lcd.print(".-.-.-.-DANGER.-.-.-.-.");  
    lcd.setCursor(0,1) ;  
lcd.print("GAS LAVEL:");  
lcd.print(Ss1);  
  
  
         
    digitalWrite(2,LOW);  
digitalWrite(3,HIGH);  
  
               
  
  
Serial.print("AT\r");  
delay(3000);  
Serial.print("AT+CMGF=1\r");  
delay(3000);  
Serial.print("AT+CMGS=\"+919566789110\"\r");  
delay(3000);  
Serial.print("\*\*LOCATION -- ABNORMAL GAS DETECTED IN TRACK");  
Serial.print("\r");    
delay(3000);    
Serial.print((char)26);                    
delay(5000);  
Serial.print("AT\r");  
delay(3000);  
Serial.print("ATD+919566789110;\r");  
delay(1000);  
  
  
      }  
       
      delay(100);  
        }

**SCREEN SHOT**

****

**CONCLUSION**

An overall conclusion IOT based toxic gas detector, or IOT technology has come a long way since it was conceptualized two decades ago. It has become more efficient, more applicable to today’s applications and smarter. The work presented in this project was directed towards pushing IOT technology to the next level. The work has presented solutions to several problems and issues that have not been addressed in previous work. The principle of operation of Operation of IOT based gas leakage and monitoring system was shown by operating the Arduino model attached with embedded system with required input and output gas level with the help of gas sensors. This results in a more efficient in operation because it is connected to a common web page specially built to notify or email the responsible authority automatically so reduces the stress of constant monitoring. The choice of using a real time gas leakage monitoring and sensing the output levels of gas has been clearly observed by the help of this system.

**REFERANCE**

[1] Soundarya, T., et al., C-leakage: Cylinder LPG Gas leakage Detection for home safety. IOSR Journal of Electronics and Communication (IOSR-JECE), 2014. 9(1): p. 53-58.

[2] ÖZMEN, Ramazan, and Mustafa Günay. "An Investigation on Deformation Behaviors of Energy Absorbers for Passenger Coaches." Periodicals of Engineering and Natural Sciences (PEN) 5, no. 3 (2017).

[3] Sharif, Md Haidar, Ivan Despot, and Sahin Uyaver. "A proof of concept for home automation system with implementation of the internet of things standards." Periodicals of Engineering and Natural Sciences (PEN) 6, no. 1 (2018): 95-106.

[4] Fraiwan, L., et al. A wireless home safety gas leakage detection system. in Biomedical Engineering (MECBME), 2011 1st Middle East Conference on. 2011. IEEE.

[5] Mustafa, K. and H. Gitano-Briggs. Liquefied petroleum gas (LPG) as an alternative fuel in spark ignition engine: Performance and emission characteristics. in Energy and Environment, 2009. ICEE 2009. 3rd International Conference on. 2009. IEEE.

[6] Fauadi, Muhammad Hafidz Fazli Md, Suriati Akmal, Mahasan Mat Ali, Nurul Izah Anuar, Samad Ramlan, Ahamad Zaki Mohd Noor, and Nurfadzylah Awang. "Intelligent Vision-based Navigation System for Mobile Robot: A Technological Review." Periodicals of Engineering and Natural Sciences (PEN) 6, no. 2 (2018): 47-57.

[7] Sabharwal, N., et al., A low cost zigbee based automatic wireless weather station with GUI and web hosting facility. 2014, ICRTEDC.

[8] Mahmood, S.N. and F.F. Hasan, Design of Weather Monitoring System Using Arduino Based Database Implementation. Journal of Multidisciplinary Engineering Science and Technology (JMEST), ISSN, 2017: p. 2458-9403.

[9] Andrade, P. B., Cruvinel, P. E., & Pe, E. A. (2018, January). Module for Virtual Calibration of Sensors of Agricultural Spraying Systems (Temperature, Pressure and flow) Using an Arduino-Based Architecture and a Controller Area Network Bus (CAN). In 2018 IEEE 12th International Conference on Semantic Computing (ICSC) (pp. 352-357). IEEE.

[10] Nasution, T.H., et al. Electrical appliances control prototype by using GSM module and Arduino. in Industrial Engineering and Applications (ICIEA), 2017 4th International Conference on. 2017. IEEE.

[11] Lapshina, P. D., Kurilova, S. P., & Belitsky, A. A. (2019, January). Development of an Arduinobased CO2 Monitoring Device. In 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (pp. 595-597). IEEE.

[12] Nograles, H.A.H., et al. Low cost Internet based wireless sensor network for air pollution monitoring using Zigbee module. in Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on. 2014. IEEE.

[13] Machappa, T., M. Sasikala, and M.A. Prasad, Design of Gas Sensor Setup and Study of Gas (LPG) Sensing Behavior of Conducting Polyaniline/Magnesium Chromate (MgCrO $ \_ {4} $) Composites. IEEE Sensors Journal, 2010. 10(4): p. 807-813.

[14] Geng, L., Li, Y., Shao, S., Liu, D., & Zhang, Q. (2018, February). The Design of the Intelligent Window Control System Based on Wireless. In 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) (pp. 294-297). IEEE.