A Report on

**"INDUSTRIAL TRAINING"**

Submitted in partial fulfilment of the requirement for the

Diploma

In

**ELECTRONICS AND COMMUNICATION ENGINEERING**

by

……

Under the guidance of

the trainers at

SV GLOBAL SERVICES PVT LTD.

Hyderabad

# ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all the people who gave us the possibility to complete this report and training at SV Global Services. It is always a pleasure to remind the fine people for their sincere guidance we received to uphold our practical as well as theoretical skills in this industrial training.

Firstly, I would like to thank all the trainers at SV Global Services who gave us all the support required to understand all the technical aspects both in theory and practice in the training.

Secondly, I would like to thank all my lecturers for guiding us throughout the industrial training whenever necessary.

Finally, we would like to extend our gratitude to our families, for their love, patience and understanding.

# DECLARATION

We sincerely declare that:

1. The Industrial training is completed.

2. The details of training and experience contained in this report describe our involvement as a trainee in the fields of Testing and Embedded systems.

3. All the information contained in this report is certain and correct to us.

NAME:

SIGNATURE:

DATE:

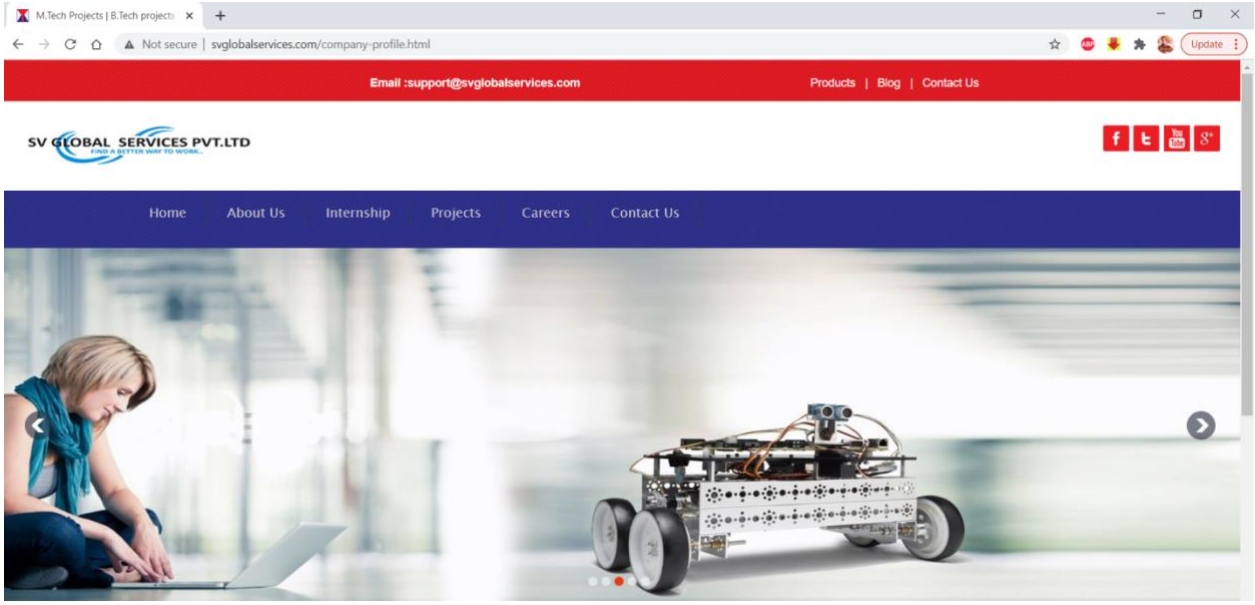
# CONTENTS

Project on “Password based door locking system using Microcontroller”…….

# INTRODUCTION

## ABOUT COMPANY:

## SV GLOBAL SERVICES OPC PVT LTD




## Vision:

Our vision is to become the employers preferred Recruitment Agency choice based on superior quality, Quantity of our 3g hr service. We also endeavor to be the preferred platform for candidates to innovate and pursue exciting careers with their ideal & dreams companies.

## Code of conduct

## Confidentiality:

We give great preference to confidentiality. Any information shared with us during any assignment remains confidential and is not disclosed to any third party. We ensure that both the parties are immediately informed of any conflict of interests or the slightest possibility of the same between clients and potential clients.

## Communication:

We maintain transparency in our communication both with client and candidates. Information to client is provided clearly if we find a particular assignment is within our capabilities/focus or not. Agreement is made about the professional fees for the executive search with the client prior to the commencement of the project.

## Quality:

Quality plays a crucial role in the services we provide. We never encourage unsolicited profiles nor send them to client from our end. From the date of joining, for one year we donâ€™t provide any openings to the candidate provided if the client requests us for outplacement of the candidate always we represent our client during salary negotiations, but being fair with the candidate at the same time

## Company Profile

SV Global Services is a Hyderabad, India based Technology Company that is into product design, services, training and custom designed projects. It is an ISO 9001-2008 certified organization which has high level expertise In cutting edge technologies including embedded software development, real time systems, RFID solutions, GSM, GPS, Zigbee and wireless factory automations. SV Global Services offers comprehensive engineering solutions based on embedded platform has developed several embedded applications for use in areas such as factory and building automation, migration and porting to different hardware and software platforms.

SV Global Services is an island of innovations, where knowledge meets curiosity, has been setting new bench marks in engineering training and solutions. Nestled in prime location in the heart of Hyderabad city, SV Global Services believes in empowering nation by technology training, creativity and freedom to ensure that learning is fun.

The team members at wine yard technologies upgrade their skill set time-to-time in various advanced technology domains to cater the industry. More than 1,00,000 students and over 400 working professionals, professors and teaching faculty from various universities are immensely benefited by workshops conducted by us on various technology domains.

# TESTING

## What is software?

Set of programs i.e. instructions to Form Applications

## Software Types

The term '**software**' refers to the set of electronic program instructions or data a computer processor reads in order to perform a task or operation. In contrast, the term '**hardware**' refers to the physical components that you can see and touch, such as the computer hard drive, mouse, and keyboard.

Software can be categorized according to what it is designed to accomplish. There are two main types of software: **systems software** and **application software**.

There are two main types of software: **systems software** and **application software**.

**Systems software's** :- These includes the programs that are dedicated to managing the computer itself, such as the **operating system**, file management utilities, and disk operating system (or DOS).

**Application software's:-** These **applications**, are often called productivity programs or end-user programs because they enable the user to complete tasks such as creating documents, spreadsheets, databases, and publications, doing online research, sending email, designing graphics, running businesses, and even playing games

## Basically Software Applications are 3 Types

- Web Based Applications
- Window based Applications
- Distributed Applications

## Different Types of Domain Technologies

- Brand New technology
- Domain Specific Technology
- Migrating to New Technology
- System is Ready Adding New Features

➢ **Software Testing :** According to IEEE It is a process of evaluating a system with respect of Manual or by automation process and verifies the difference between the expected and the actual result. Testing analyze the program with the intent of finding problem and errors which measures system functionality and quality.

or

The process of executing a system with the intent of finding defects. To ensure the application Quality with Zero Defects

or

Verification and Validation is also called as testing.

Testing = Verification + Validation

Building the product right

Building the right product

**Re- Testing:** Testing functionality again or testing functionality repetitively is called

re-testing.

Re-testing comes in the following 2 scenarios.

1) Testing a functionality with multiple inputs to confirm if the business validations are implemented or not

2) Testing a functionality on the modified build to confirm the bug fixers are made correctly or not.

**Regression Testing:** It is process of identifying various features in the modified build where there is a chance of getting affected and retesting these features.

The new functionalities added to the existing system or modifications made to the existing system or the bug fixes may introduce side-effects.

Regression testing is helpful to identify these side effects.

## Various System Testing Types

Usability Testing

**Usability Testing** is a black box **testing** technique. **Usability testing** also reveals whether users feel comfortable with your application or Web site according to different parameters – the flow, navigation and layout, speed and content – especially in comparison to prior or similar applications.

Functional Testing

Functional Testing(Covers +ve, -Ve testing)

Validation which Should meet Business Requirement

Validation which is going to fulfil End-users needs

Performance Testing

**Performance testing** is **testing** that is performed, to determine how fast some aspect of a system performs under a particular workload.

It can serve different purposes like it can demonstrate that the system Meets **performance** criteria.

Security Testing

**Security testing** is a **testing** technique to determine if an information system protects data and maintains functionality as intended.

It also aims at verifying 6 basic principles as listed below: Confidentiality. Integrity.

## Testing Methodologies:

**White box Testing**→ Structural(syntax and run time errors)

**Black box Testing**--→ Functional Testing(Covers +ve, -Ve testing)

Validation which Should meet Business Requirement Process

Validation which is going to fulfil End-users satisfaction

**White box Testing**: Testing conducted on the source code by developers to check does the source code is working as expected or not is called white box testing.

**Note:** Unit testing and integration testing is collectively called white box testing.

## What is the need of white box testing?

As the source code is visible, finding and rectifying the problems is easy for developers. The defects that are identified in white box testing are very economical to resolve. To reduce the defects as early as possible white box testing is helpful.
To ensure 100% code coverage.
**White box testing is also called as Glass box, Structural or Clear Box Testing.**

**Black box Testing:** Testing is conducted on the application by test engineers or by domain experts to check whether the application is working according to customer requirements. It is also Called as Functional Testing

## What is the need of Black Box Testing?

1) White box testing conducted by developer in a technical perception where as black box testing is conducted by test engineers with end-user perception.
2) Programmers will conduct white box testing in a positive perception whereas tester will conduct black box testing with a negative perception where there is a more chance of finding more defects
   The more defects you identify results in a quality system.

3) White box testing will not cover non functional areas. As Functional requirements are also very important for production system those are covered in black box testing.
4) The objective of white box testing is 100% code coverage whereas the objective of black box testing is 100% customer business requirement coverage.
5) Black Box Testing = System Testing + User Acceptance Testing which is called as requirement Based Testing (or) Specification Based Testing.

**GREY BOX TESTING**: It is method of testing in which one will perform testing on both the functional part as well as structural part of on application. ➢ Usually the Test engineer'swho has the knowledge of structural part will perform.

**Testing:** It is a process of verifying are we developing the right product or not and also validating does the developed product is right or not.
**Software testing = Verification + Validation.**

| Testing = | Verification | + | Validation |
|---|---|---|---|
| | Building the product right | | Building the right product |

## What is Verification?

It is a process of verifying: Are we developing the right product or not. Known as static testing.

## What is Validation?

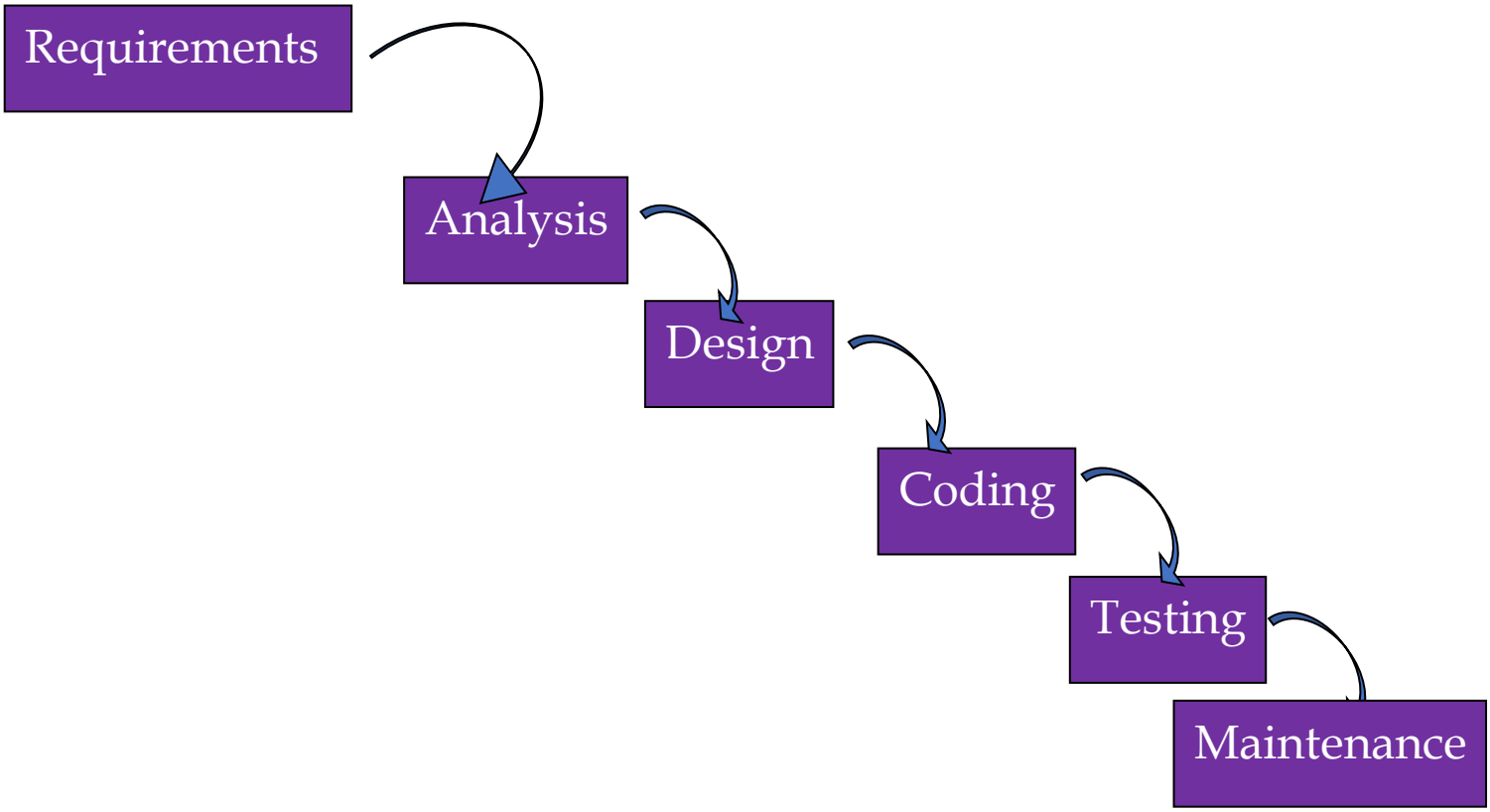
It is a process of validating: Does the developed product is right or not. Also called as dynamic testing.

## Software Development models

- Waterfall model or **Linear sequential model**
- Incremental model
- Prototype Model
- RAD Model
- Spiral Model
- V-Model

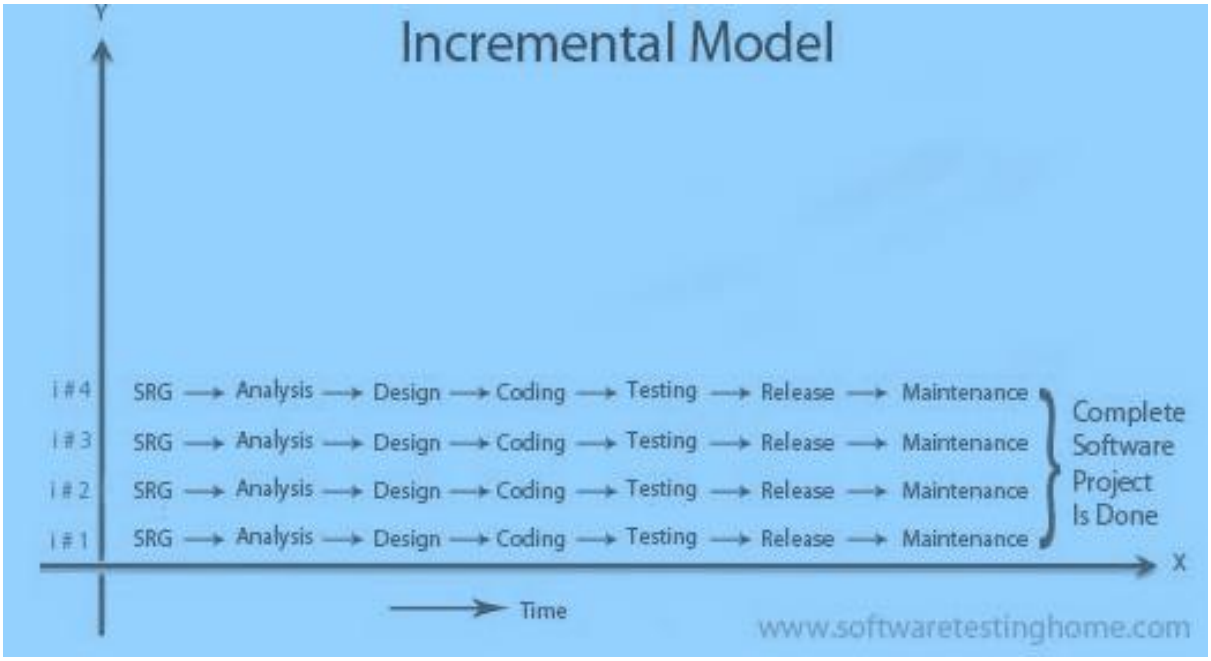
- Agile Methdology

## WATER FALL MODEL



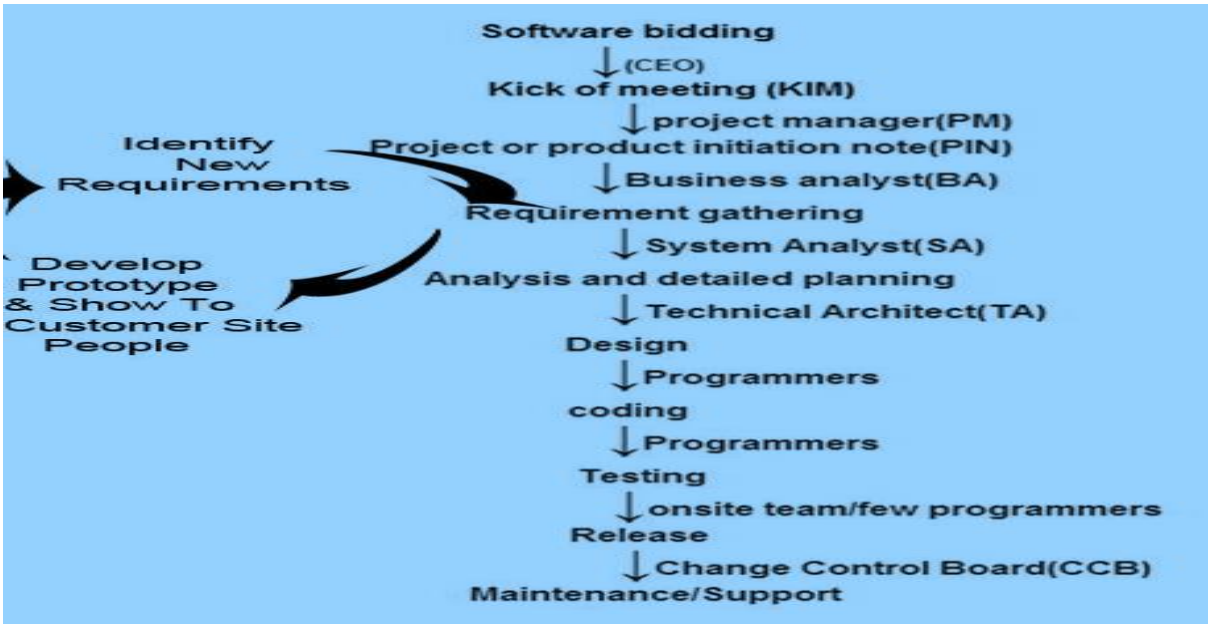

- SDLC Waterfall model is used when: Requirements are stable and not changed frequently.
- An application is small.
- There is no requirement which is not understood or not very clear.
- The environment is stable
- The tools and techniques used is stable and is not dynamic
- Resources are well trained and are available

## Incremental Model




- When to use the Incremental model:
- 1. This model can be used when the requirements of the complete system are clearly defined and understood.
- 2. Major requirements must be defined; however, some details can evolve with time.
- 3. There is a need to get a product to the market early.
- 4. A new technology is being used
- 5. Resources with needed skill set are not available 6. There are some high risk features and goals

## Prototype model

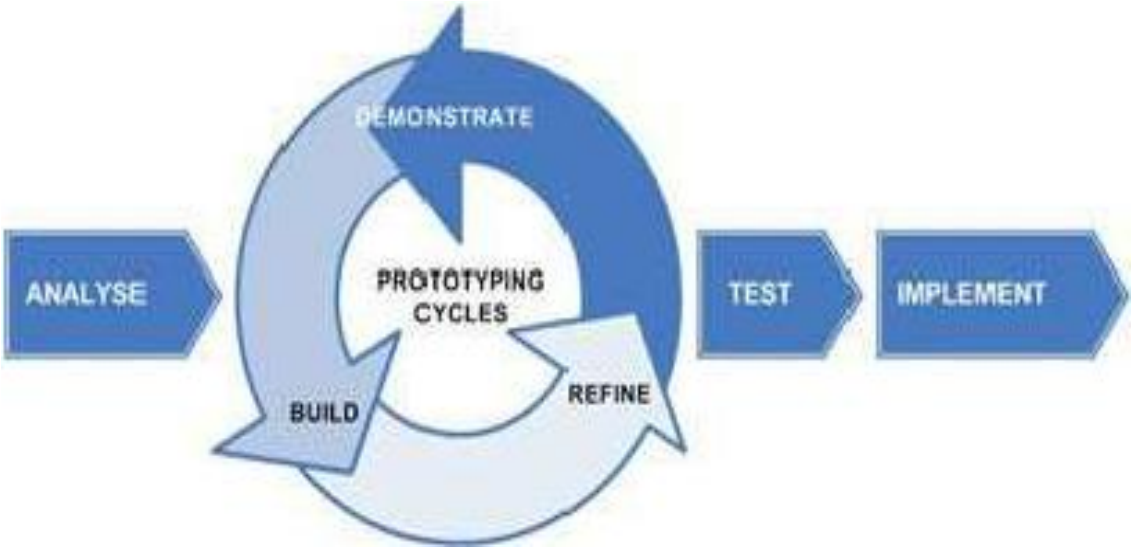


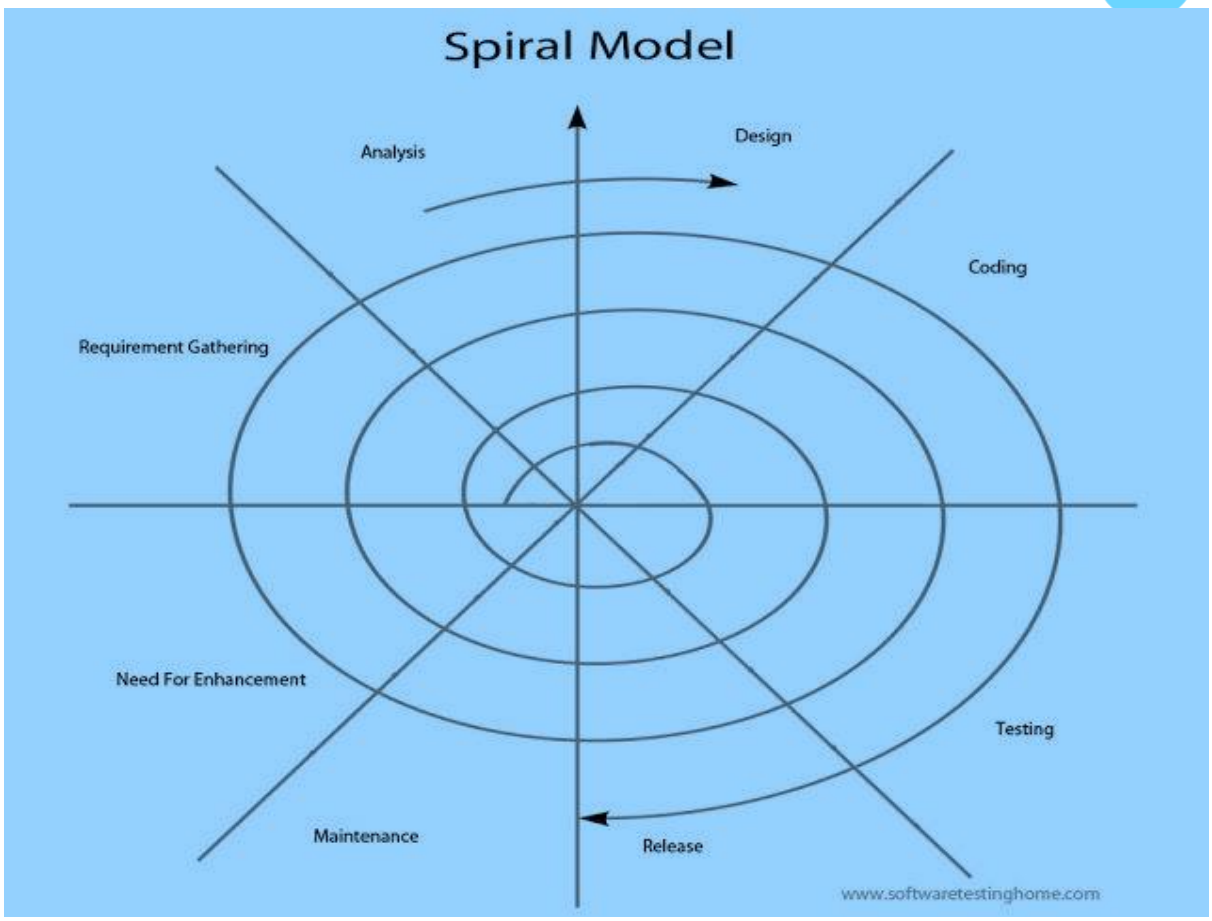

- When to use Prototype model:

- 1. Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- 2. Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- 3. Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

## Rapid Application development model

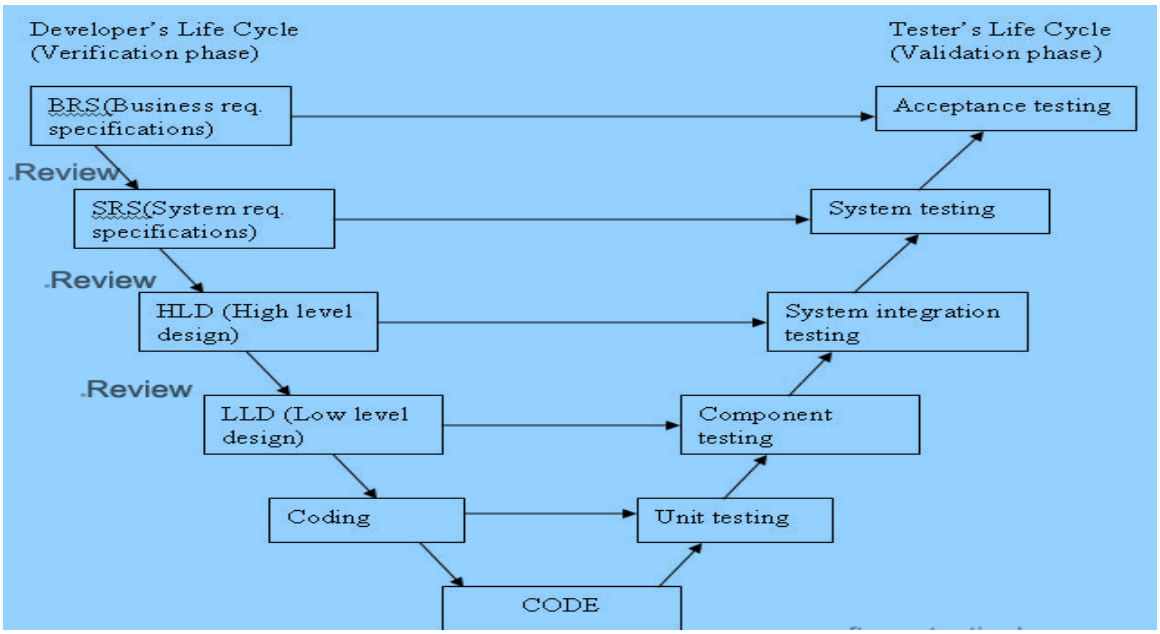
RAD model is well suitable for developing big size projects in a **short span of time .**In RAD Model a big project divided into modules ,then every module will be considered as a separate project and dedicated teams scheduled to implement these modules. Once all modules are constructed, these modules combined together tested, then deliver to the client. As the project getting implemented very quickely.This model called as RAD Model






- When to Use Spiral model? - Spiral model is used in the following scenarios: -
- When the project is large. –
- Where the software needs continuous risk evaluation.
- - Requirements are a bit complicated and require continuous clarification. –
- Software requires significant changes. - Where enough time frame is their to get end user feedback.
- - Where releases are required to be frequent.

## V Model




## AGILE METHODOLOGY

In English, Agile means 'ability to move quickly and easily' and responding swiftly to change – this is a key aspect of Agile software development as well.

AGILE is a methodology that promotes **continuous iteration** of development and testing throughout the software development life cycle of the project. Both development and testing activities are concurrent

### Scrum

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team based development environment. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members). It consists of three roles, and their responsibilities are explained as follows:

### Scrum Master

Master is responsible for setting up the team, sprint meeting

### Product owner

The Product Owner is responsible for the delivery of the functionality at each iteration

### Scrum Team

Team manages its own work and organizes the work to complete the sprint or cycle

**Test Scenario:** An item or a feature or a functionality to be tested in the application under test is called test scenario.

or

Test scenarios are nothing but one line pointers of 'what to test' for a certain functionality.

or

All the possible areas to be tested

Example: Few Test scenarios for Gmail Application login, Inbox, Sent, Trash, Spam
Few Scenarios for Danske bank like Admin,Banker,Customer Modules

**Test Case:** A Test Case is set of preconditions steps to be followed with input data and expected behavior to validate functionality a system.

Or

**Test case is a single document which describes the description expected, result and actual result of the application which is used to test the application and find the defects.**

Or

Test Case :All the Possible conditions to be Tested

Two types of functional test cases:

1) Positive test cases 2) Negative test cases

# EMBEDDED SYSTEMS

## INTRODUCTION:

An embedded system is a combination of computer hardware and software designed for a specific function or functions within a larger system. The systems can be programmable or with fixed functionality. Industrial machines, consumer electronics, agricultural and process industry devices, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines and toys, as well as mobile devices, are possible locations for an embedded system.

## CHARACTERISTICS OF EMBEDDED SYSTEMS

The main characteristic of embedded systems is that they are task specific. They perform a single task within a larger system. For example, a mobile phone is not an embedded system, it is a combination of embedded systems that together allow it to perform a variety of general-purpose tasks. The embedded systems within it perform specialized functions. For example, the GUI performs the singular function of allowing the user to interface with the device. In short, they are programmable computers, but designed for specific purposes, not general ones.

## ADDITIONALLY, EMBEDDED SYSTEMS CAN INCLUDE THE FOLLOWING

## CHARACTERISTICS:

• comprised of hardware, software and firmware;

• embedded in a larger system to perform a specific function as they are built for specialized tasks within the system, not various tasks;

• either microprocessor-based or microcontroller-based -- both are integrated circuits that give the system compute power;

• often used for sensing and real-time computing in internet of things (IoT) devices -- devices that are internet-connected and do not require a user to operate;

• vary in complexity and in function, which affects the type of software, firmware and hardware they use; and

• often required to perform their function under a time constraint to keep the larger system functioning properly.

## EMBEDDED SYSTEMS VARY IN COMPLEXITY, BUT GENERALLY CONSIST OF THREE MAIN ELEMENTS:

• **Hardware:** The hardware of embedded systems is based around microprocessors and microcontroller. Microprocessors are very similar to microcontrollers, and generally refer to a CPU that is integrated with other basic computing components such as memory chips and digital signal processors . Microcontrollers have those components built into one chip.

• **Software:** Software for embedded systems can vary in complexity. However, industrial-grade microcontrollers and embedded IoT systems generally run very simple software that requires little memory.

# BASIC BLOCK DIAGRAM OF EMBEDDED SYSTEM:

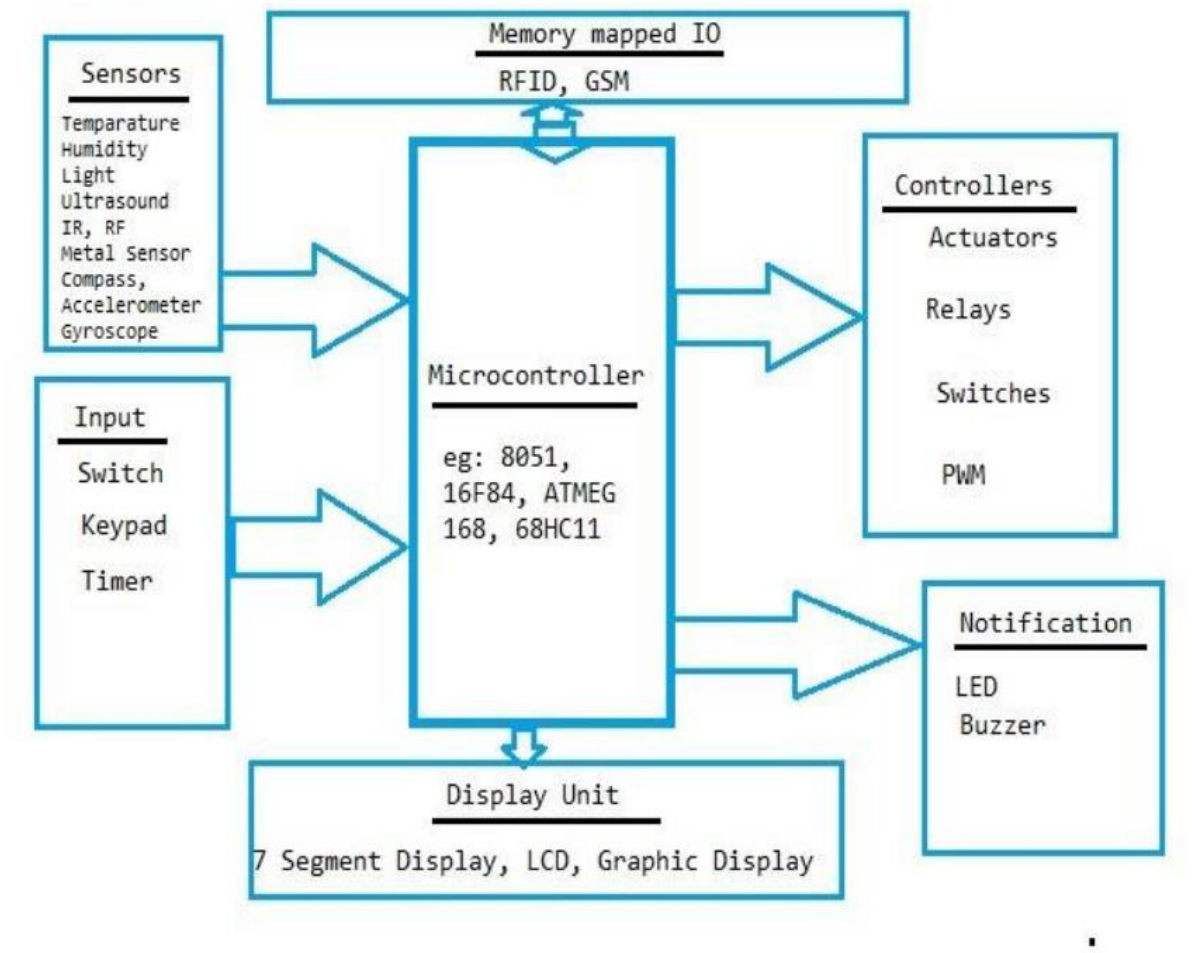



**Fig : Block Diagram Of Embedded System**

# APPLICATIONS OF EMBEDDED SYSTEMS:

Embedded systems are used in a variety of technologies across industries. Some examples include:

• **Automobiles.** Modern cars commonly consist of many computers (sometimes as many as 100), or embedded systems, designed to perform different tasks within the vehicle. Some of these systems perform basic utility function and others provide entertainment or user-facing functions. Some embedded systems in consumer vehicles include cruise control, backup sensors, suspension control, navigation systems and airbag systems.

• **Mobile phones.** These consist of many embedded systems, including GUI software and hardware, operating systems, cameras, microphones and USB I/O modules.

• **Industrial machines.** They can contain embedded systems, like sensors, and can be embedded systems themselves. Industrial machines often have embedded automation systems that perform specific monitoring and control functions.

• **Medical equipment.** These may contain embedded systems like sensors and control mechanisms.

Medical equipment, such as industrial machines, also must be very user-friendly, so that human health isn't jeopardized by preventable machine mistakes. This means they'll often include a more complex OS and GUI designed for an appropriate UI.

## ADVANTAGES OF EMBEDDED SYSTEMS

• The embedded system is easy for mass production.

• The embedded system is highly reliable.

• It has very few interconnections.

• The embedded system is small in size

• The embedded system has less expensive

.• It has fast operation.

• It has improved product quality.

• It optimizes use of system resources.

• It has low power operation.

## DISADVANTAGES OF EMBEDDED SYSTEM

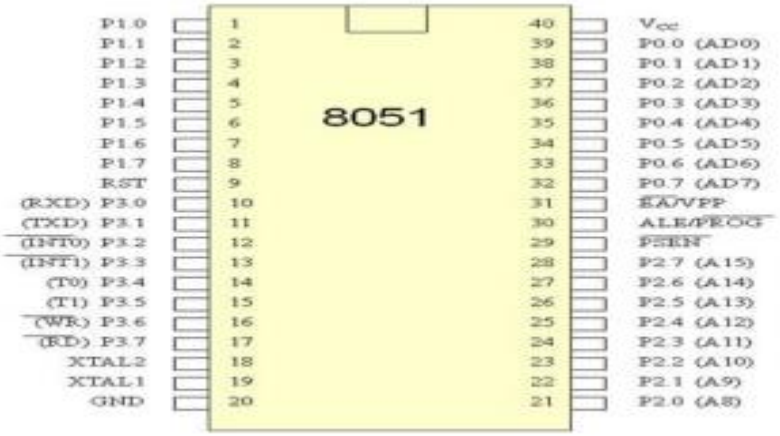
•The embedded systems are hard for maintenance as it is use and throw device.

• It has no technological improvement.

• Less power supply durability if it is battery operated.

• It has hard to take backup of embedded files.

# PIN DIAGRAM




# PINOUT DESCRIPTION

| | |
|---|---|
| **Pins 1- 8** | **PORT 1.** Each of these pins can be configured as an input or an output. |
| Pin 9 | RESET. A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning. |
| **Pins10- 17** | **PORT 3.** Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions |
| **Pin 10** | **RXD.** Serial asynchronous communication input or Serial synchronous communication output |

| | |
|---|---|
| **Pin 11** | **TXD.** Serial asynchronous communication output or Serial synchronous communication clock output |
| **Pin 12** | **INT0.**External Interrupt 0 input |
| **Pin 13** | **INT1.** External Interrupt 1 input |
| **Pin 14** | **T0**. Counter 0 clock input |
| **Pin 15** | **T1**. Counter 1 clock input |
| **Pin 16** | **WR.** Write to external (additional) RAM |
| **Pin 17** | **RD.** Read from external RAM |
| **Pin 18,19** | **XTAL2, XTAL1**. Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins |
| **Pin 20** | **GND.** Ground. |
| **Pin 21-28** | **Port 2**. If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs. |
| **Pin 29** | **PSEN.** If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory. |
| **Pin 30** | **ALE**. Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external latch latches the |

| | |
|---|---|
| | state of P0 and uses it as a memory chip address. Immediately after that, the ALE pin is returned its previous logic state and P0 is now used as a Data Bus. |
| **Pin 31** | **EA.** By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists). |
| **Pin 32-39** | **PORT 0.** Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0). |
| **Pin 40** | **VCC.** +5V power supply. |

# EMBEDDED DESIGN PROCESS

## EMBEDDED SYSTEM OVERVIEW

In the earliest years of computers in the 1930–40s, computers were sometimes dedicated to a single task, but were far too large and expensive for most kinds of tasks performed by embedded computers of today. Over time however, the concept of programmable controllers evolved from traditional electromechanical sequencers, via solid state devices, to the use of computer technology.

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

## TOOLS

Embedded development makes up a small fraction of total programming. There's also a large number of embedded architectures, unlike the PC world where 1 instruction set rules, and the Unix world where there's only 3 or 4 major ones. This means that the tools are more expensive. It also means that they're lower featured, and less developed. On a major embedded project, at some point you will almost always find a compiler bug of some sort. Debugging tools are another issue. Since you can't always run general programs on your embedded processor, you can't always run a debugger on it. This makes fixing your program difficult. Special hardware such as JTAG ports can overcome this issue in part. However, if you stop on a breakpoint when your system is controlling real world hardware (such as a motor), permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

## RESOURCES

Embedded development makes up a small fraction of total programming. There's also a large number of embedded architectures, unlike the PC world where 1 instruction set rules, and the Unix world where there's only 3 or 4 major ones. This means that the tools are more expensive. It also means that they're lower featured, and less developed. On a major embedded project, at some point you will almost always find a compiler bug of some sort. Debugging tools are another issue. Since you can't always run general programs on your embedded processor, you can't always run a debugger on it. This makes fixing your program difficult. Special hardware such as JTAG ports can overcome this issue in part. However, if

you stop on a breakpoint when your system is controlling real world hardware (such as a motor), permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

## DEBUGGING

Embedded debugging may be performed at different levels, depending on the facilities available. From simplest to most sophisticate they can be roughly grouped into the following areas:

- Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
- External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger which even works for heterogeneous multi core systems.
- An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface. This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
- An in-circuit emulator replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
- A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified and allowing debugging on a normal PC.
- Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as assembly code or source-code.

Because an embedded system is often composed of a wide variety of elements, the debugging strategy may vary. For instance, debugging a software(and microprocessor) centric embedded system is different from debugging an embedded system where most of the processing is performed by peripherals (DSP, FPGA, co-processor). An increasing number of embedded systems today use more than one single processor core. A common problem with multi-core development is the proper synchronization of software execution. In such a case, the embedded system design may wish to check the data traffic on the busses between the processor cores, which requires very low-level debugging, at signal/bus level, with a logic analyzer, for instance.

## RELIABILITY

Embedded systems often reside in machines that are expected to run continuously for years without errors and in some cases recover by themselves if an error occurs. Therefore the software is usually developed and tested more carefully than that for personal computers, and unreliable mechanical moving parts such as disk drives, switches or buttons are avoided.

Specific reliability issues may include:

- The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
- The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals, engines on single-engine aircraft.
- The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.

A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and also soft errors in the hardware:

- Watchdog timer that resets the computer unless the software periodically notifies the watchdog
- Subsystems with redundant spares that can be switched over to
- software "limp modes" that provide partial function
- Designing with a Trusted Computing Base (TCB) architecture[6] ensures a highly secure & reliable system environment
- An Embedded Hypervisor is able to provide secure encapsulation for any subsystem component, so that a compromised software component cannot interfere with other subsystems, or privileged-level system software. This encapsulation keeps faults from propagating from one subsystem to another, improving reliability. This may also allow a subsystem to be automatically shut down and restarted on fault detection.
- Immunity Aware Programming

## CHARACTERISTICS OF EMBEDDED SYSTEMS

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete

device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites. (Each radar probably includes one or more embedded systems of its own.)

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which don't expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded". A modern example of embedded system is shown in fig: 3.6.1.

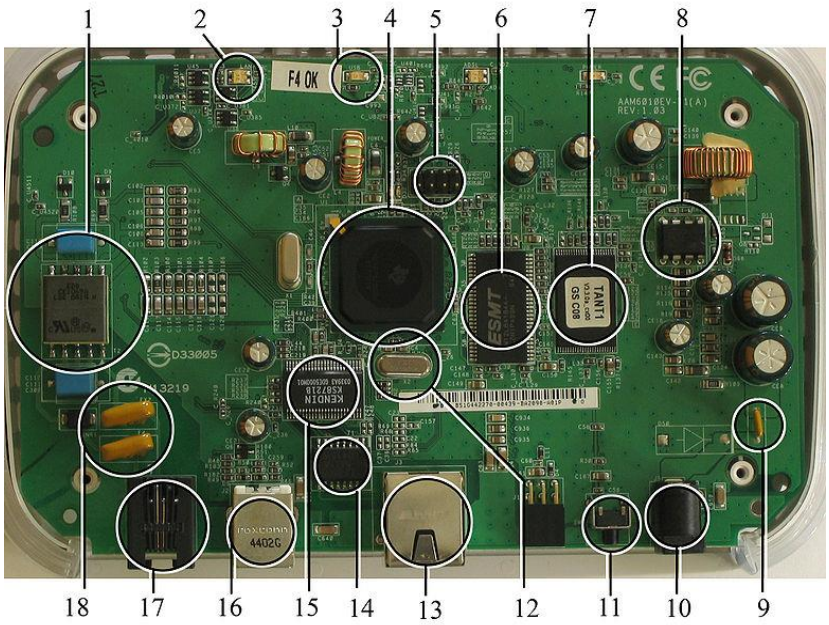



**Fig :A modern example of embedded system**

Labeled parts include microprocessor (4), RAM (6), flash memory (7).Embedded systems programming is not like normal PC programming. In many ways, programming for an embedded system is like programming PC 15 years ago. The hardware for the system is usually chosen to make the device as cheap as possible. Spending an extra dollar a unit in order to make things easier to program can cost millions. Hiring a programmer for an extra month is cheap in comparison. This means the programmer must make do with slow processors and low memory, while at the same time battling a need for efficiency not seen in most PC applications. Below is a list of issues specific to the embedded field.

## APPLICATIONS OF EMBEDDED SYSTEMS:

### Consumer applications:

At home we use a number of embedded systems which include microwave oven, remote control, vcd players, dvd players, camera etc….

**Fig: Automatic coffee makes equipment**

## Office automation:

We use systems like fax machine, modem, printer etc…

**Fig: Fax machine**

**Fig: Printing machine**

## Industrial automation:

Today a lot of industries are using embedded systems for process control. In industries we design the embedded systems to perform a specific operation like monitoring temperature, pressure, humidity ,voltage, current etc.., and basing on these monitored levels we do control other devices, we can send information to a centralized monitoring station.


**Fig: Robot**

In critical industries where human presence is avoided there we can use robots which are programmed to do a specific operation.

## Computer networking:

Embedded systems are used as bridges routers etc..


**Fig: Computer networking**

## Tele communications:

Cell phones, web cameras etc.


**Fig: Cell Phone** **Fig: Web camera**

# MICROPROCESSORS AND MICROCONTROLLERS

| MICROPROCESSOR | MICROCONTROLLER |
|---|---|
| Microprocessor contains ALU, General purpose registers, stack pointer, program counter, clock timing circuit, interrupt circuit | Microcontroller contains the circuitry of microprocessor, and in addition it has built in ROM, RAM, I/O Devices, Timers/Counters etc. |
| It has many instructions to move data between memory and CPU | It has few instructions to move data between memory and CPU |
| Few bit handling instruction | It has many bit handling instructions |
| Less number of pins are multifunctional | More number of pins are multifunctional |
| Single memory map for data and code (program) | Separate memory map for data and code (program) |
| Access time for memory and IO are more | Less access time for built in memory and IO. |
| Microprocessor based system requires additional hardware | It requires less additional hardwares |
| More flexible in the design point of view | Less flexible since the additional circuits which is residing inside the microcontroller is fixed for a particular microcontroller |
| Large number of instructions with flexible addressing modes | Limited number of instructions with few addressing modes |

## HARVARD & VON- NEUMANN CPU ARCHITECTURE

| **Von-Neumann (Princeton architecture)** | **Harvard architecture** |
|---|---|
| It uses single memory space for both instructions and data. | It has separate program memory and data memory |
| It is not possible to fetch instruction code and data | Instruction code and data can be fetched simultaneously |
| Execution of instruction takes more machine cycle | Execution of instruction takes less machine cycle |
| Uses CISC architecture | Uses RISC architecture |
| Instruction pre-fetching is a main feature | Instruction parallelism is a main feature |
| Also known as control flow or control driven computers | Also known as data flow or data driven computers |
| Simplifies the chip design because of single memory space | Chip design is complex due to separate memory space |
| Eg. 8085, 8086, MC6800 | Eg. General purpose microcontrollers, special DSP chips etc. |

## RISC AND CISC CPU ARCHITECTURES

Microcontrollers with small instruction set are called reduced instruction set computer (RISC) machines and those with complex instruction set are called complex instruction set computer (CISC). Intel 8051 is an example of CISC machine whereas microchip PIC 18F87X is an example of RISC machine.

| RISC | CISC |
|---|---|
| Only load/store instructions are used to access memory | In additions to load and store instructions, memory access is possible with other instructions also |
| Instructions executed by hardware | Instructions executed by the micro program |
| Fixed format instruction | Variable format instructions |
| Few addressing modes | Many addressing modes |
| Few instructions | Complex instruction set |
| Most of the have multiple register banks | Single register bank |
| Highly pipelined | Less pipelined |
| Complexity is in the compiler | Complexity in the microprogram |

# THE 8051 ARCHITECTURE

## Introduction

## Salient features of 8051 microcontroller are given below.

• Eight bit CPU

• On chip clock oscillator

• 4Kbytes of internal program memory (code memory) [ROM]

• 128 bytes of internal data memory [RAM]

• 64 Kbytes of external program memory address space.

• 64 Kbytes of external data memory address space.

• 32 bi directional I/O lines (can be used as four 8 bit ports or 32 individually addressable I/O lines)

• Two 16 Bit Timer/Counter :T0, T1

• Full Duplex serial data receiver/transmitter Four Register banks with 8 registers in each bank.

• Sixteen bit Program counter (PC) and a data pointer (DPTR)

• 8 Bit Program Status Word (PSW)

• 8 Bit Stack Pointer

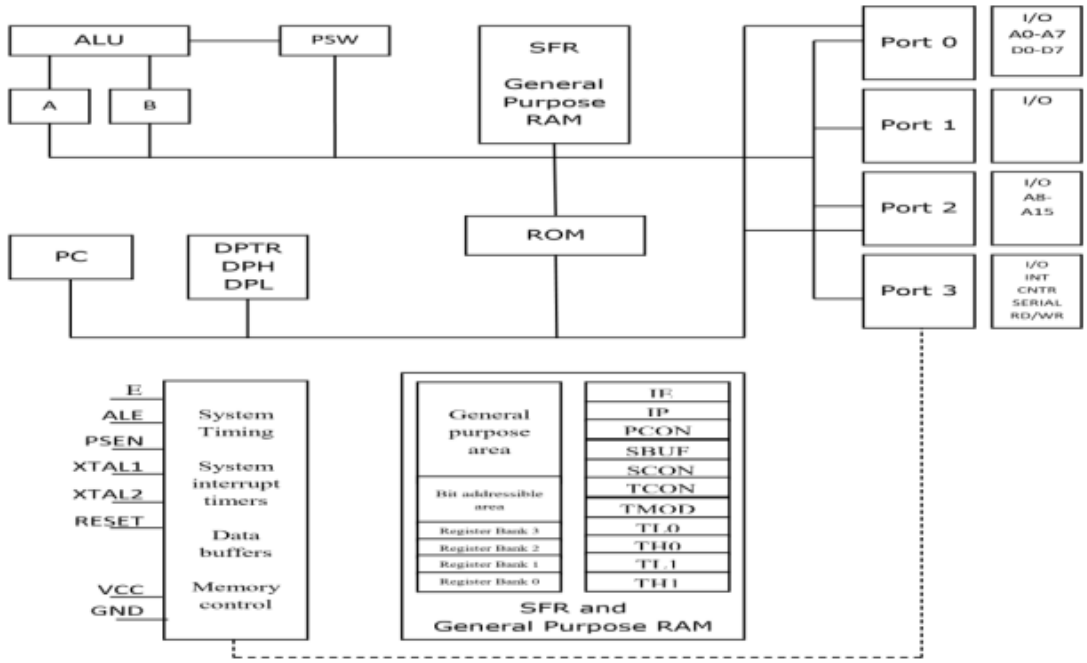
• Five vector interrupt structure (RESET not considered as an interrupt.)

• 8051 CPU consists of 8 bit ALU with associated registers like accumulator ‘A’ , B register, PSW, SP, 16 bit program counter, stack pointer.

• ALU can perform arithmetic and logic functions on 8 bit variables.

• 8051 has 128 bytes of internal RAM which is divided into o Working registers [00 – 1F] o Bit addressable memory area [20 – 2F] o General purpose memory area (Scratch pad memory) [30-7F]

## The 8051 architecture diagram




8051 has 4 K Bytes of internal ROM. The address space is from 0000 to 0FFFh. If the program size is more than 4 K Bytes 8051 will fetch the code automatically from external memory. Accumulator is an 8 bit register widely used for all arithmetic and logical operations. Accumulator is also used to transfer data between external memory. B register is used along with Accumulator for multiplication and division. A and B registers together is also called MATH registers. PSW (Program Status Word). This is an 8 bit register which contains the arithmetic status of ALU and the bank select bits of register banks.

CY - carry flag

AC - auxiliary carry

F0 - available to the user for general purpose

RS1,RS0 - register bank select bits

OV - overflow

P - parity Stack Pointer (SP) – it contains the address of the data item on the top of the stack. Stack may reside anywhere on the internal RAM. On reset, SP is initialized to 07 so that the default stack will start from address 08 onwards.

Data Pointer (DPTR) – DPH (Data pointer higher byte), DPL (Data pointer lower byte). This is a 16 bit register which is used to furnish address information for internal and external program memory and for external data memory.

Program Counter (PC) – 16 bit PC contains the address of next instruction to be executed. On reset PC will set to 0000. After fetching every instruction PC will increment by one.

# Project on

# Password Based Door Locking System Using Microcontroller

## Abstract:

The purpose of this project is to provide security at (house, ATM, office etc.) in this system the user will have to register a unique password. The information will be stored in data base. Whenever the right password will be received, the controller will accordingly give instruction to dc motor. Dc motor will perform the action on door unlocking. We want to utilize the electronic technology to build an integrated and fully customized home security system at a reasonable cost.

Many times we forgot to carry the key of our home. Or sometimes we come out of our home and door latch closes by mistake. In these cases it is really difficult to get inside the house. This project is designed to solve this purpose. Main concept behind this project is of a door-latch opening using a password entered through keypad. As well as turning on the Buzzer when password is entered wrong. Today people are facing more problems about security in all over world, nowadays security is the most essential issue everywhere in the world so security of everything gains higher and higher importance in recent years. The main component in the circuit is 8051 microcontroller. Here, 4*4 keypad is used to enter the password. The entered password is compared with the predefined password. If it is correct password, the system opens the door by rotating door motor and displays the status of door on LCD. If the password is wrong then door remains closed and displays —password is wrong on LCD.It can be used at organizations to ensure authorized access to highly secured places. With a slight modification by replacing the motor driver with arelay driver, this circuit can be used to control the switching of loads through code. This circuit can be also modified by using EEPROM chip interfaced to the microcontroller and store the entered password in the chip. Such an automatic lock system consists of electronic control assembly which controls the output load through a password. This output load can be a motor or a lamp or any other mechanical/electrical load.

## Block Diagram:

The microcontroller based door locker is an access control system that allows only authorized person to access a restricted area. The system is fully controlled by the 8 bit microcontroller 8051 which has a 2Kbytes of ROM for the program memory. The password is stored in the EPROM so that we can change it at any time. The system has a keypad by which the password can be entered through it. When the entered password equals with the password stored in the memory then the relay gets on and so that the door is opened. If we entered a wrong password then the alarm is switched on. An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

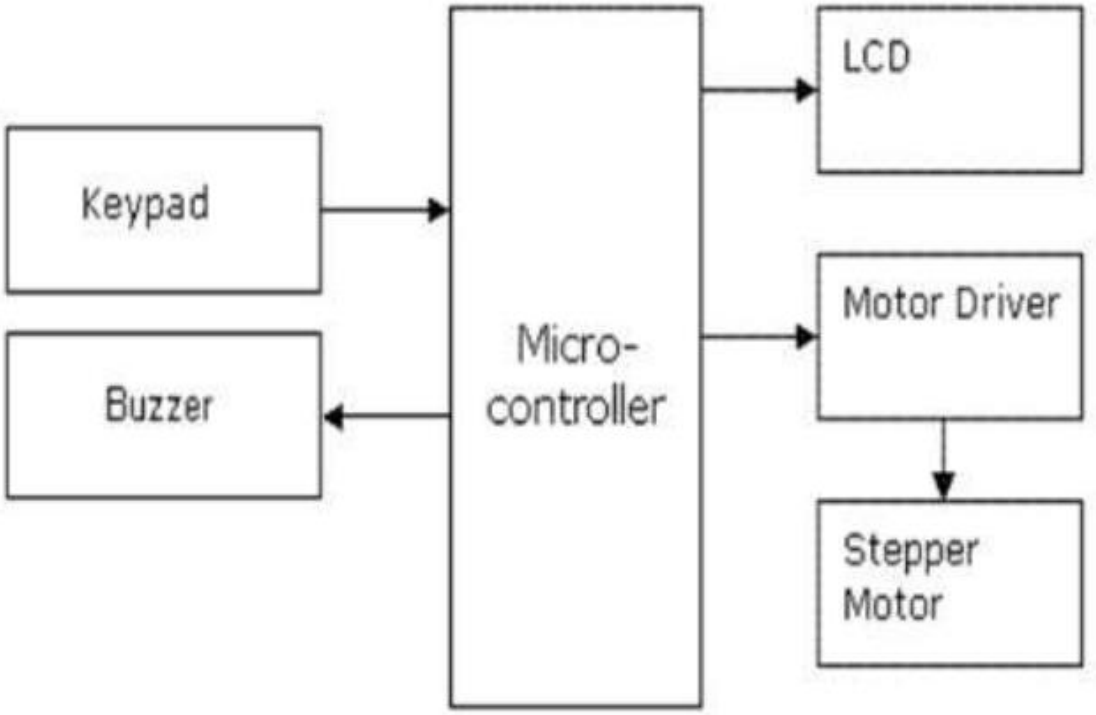



**Figure 1:** Block Diagram of password based door locking system

Components Required

Hardware Requirements

• 8051 Microcontroller

• 8051 Development Board

• 8051 Programmer

• 4×4 Matrix Keypad

• 16×2 LCD

• L293D Motor Driver Board

• DC Motor

• 10KΩ Potentiometer

• Connecting wires

• Power Supply

• If 8051 Development Board is not used, then the following components are needed.

• 11.0592 MHz Quartz Crystal

• 2 x 33pF Ceramic Capacitors

• 2 x 10 KΩ Resistor (1/4 Watt)

• 10 µF Capacitor (Polarized)

• Push Button

• 2 x 1 KΩ Resistors (for pull up)

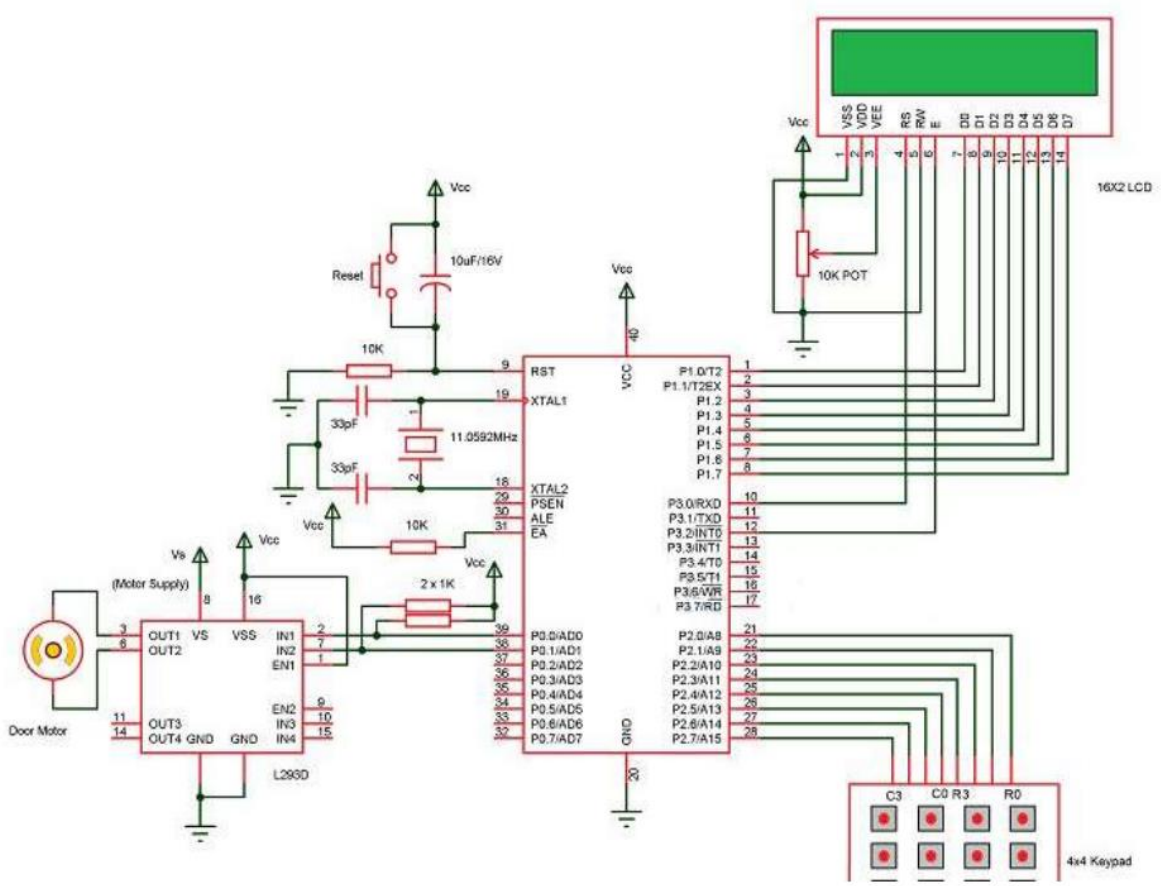
Software Requirements

• Keil µVision IDE

• Willar Programmer

• Proteus (for circuit diagram and simulation)

## Circuit Diagram:




The main component in the circuit is 8051microcontroller. Here, 4×3 keypad is used to enter the password. The entered password is compared with the predefined password. If it is correct password, the system opens the door by rotating door motor and displays the status of door on LCD.If the password is wrong then door remains closed and displays —pwd is wrong و on LCD Its design and working are very interesting and easy to implement. If you are interested to get detailed information about its design, working and applications, read the post Electronic Code Lock System using 8051 Microcontroller. Traditional lock systems using mechanical lock and key mechanism are being replaced by new advanced techniques of locking system. These techniques are an integration of mechanical and electronic devices and highly intelligent. One of the prominent features of these innovative lock systems is their simplicity and high efficiency.

## Working:

The total functioning of the—CODE LOCK SYSTEM is based on the software program which is burn inside the microcontroller IC 8051.The at89c51 IC is heart of the given circuitry because this IC is programmable 40pin dip IC in which we burn the program in rom. This IC has a 32 input lines through which we take the output pin no 9 is used for reset the microcontroller and make it in a initial condition pin no 31 is enable pin, it required low pulse for activating the microcontroller depends on the crystal connected to the xtal1 & xtal2. pin no 18&19 is used for providing the VCC of +5v pin 20 is grounded.

The operating frequency of the controller is set by the external oscillator of crystal oscillator of crystal having frequency of 12MHZ. Capacitor having 33pF in parallel connection with ground is for the proper undammed frequency. Reset for the controller is normally ground and 10uf 10v electrolytic capacitor is also connected on reset for the slowly on and off Volume 2 | Issue 3| MayJune-2017| www.ijsrcseit.com431to the controller during the supply is turning on and off. The keypad used to give input signal is been interfaced with microcontroller are port0 (p0.1-p0.7).The controller fetches the hex code according to the instruction. The LCD is used for display device it is a 16 slots device usually used to show output status from the microcontroller . The output signal which be fetched by relay status followed by on/off status of electronic lock.

## Result and Simulation

• When it is entered a 4 digit password by the user it will display on LCD as ****.Therefore anyone else can’t see what the user enters.

• If itis the correct password, LCD displaying a message —Well come and the door will be opened. after 1minute time door is locked automatically.

• If he is entered password incorrectly LCDdisplaying —password error.

• After openingthe door if user wants to change his password, after pressing —0 key and giving user id user can change his password.

• If user wants to add more people to the system after opening the door pressing —# key, user can add more users. System will give user id to each password.

Simulation of project is performed on PROTEUS and the code was written Kiel software. Code for the microcontroller to run DC motors IC (L293D) is written. In the simulation the relevant data to the Microcontroller is send through keypad.

The Microcontroller processed the data and sent the information to the Actuator IC (L293D). The Actuator IC upon receiving information showedresponse by driving the DC motors.

Source Code:

```
#include <reg51.h>

// connected pins

// keypad rows

sbit keyrow1 = P2 ^ 0;

sbit keyrow2 = P2 ^ 1;

sbit keyrow3 = P2 ^ 2;

sbit keyrow4 = P2 ^ 3;

//keypad column

sbit keycolumn1 = P3 ^ 0;

sbit keycolumn2 = P3 ^ 1;

sbit keycolumn3 = P3 ^ 2;

// motor pins

sbit motorpin1 = P3 ^ 3;

sbit motorpin2 = P3 ^ 4;

// led pins

sbit rs = P3 ^ 5;
```

```
sbit rw = P3 ^ 6;
sbit en = P3 ^ 7;
//functions
void lcdcmd(unsigned char);
void lcddat(unsigned char);
void lcddisplay(unsigned char *q);
char keypad();
void check();
void delay(unsigned int);
unsigned char pin[] = {"12345"};
unsigned char Epin[5];
// main function
void main()
{
 lcdcmd(0x0F); //decimal value: 15
 lcdcmd(0x38); //decimal value: 56
 lcdcmd(0x01); //decimal value: 1
 while (1)
 {
 unsigned int i = 0;
 lcdcmd(0x80); //decimal value: 128
 lcddisplay("ENTER PIN NUMBER");
 delay(1000);
 lcdcmd(0xc0); //decimal value: 192
```

```
 while (pin[i] != '\0')
 {
 Epin[i] = keypad();
 delay(1000);
 i++;
 }
 check();
 }
}
//delay function
void delay(unsigned int j)
{
 int a, b;
 for (a = 0; a < j; a++)
 {
 for (b = 0; b < 10; b++)
 ;
 }
}
// lcd commands functions
void lcdcmd(unsigned char A)
{
 P1 = A;
 rs = 0;
```

```
 rw = 0;
 en = 1;
 delay(1000);
 en = 0;
}
//lcd data function
void lcddat(unsigned char i)
{
 P1 = i;
 rs = 1;
 rw = 0;
 en = 1;
 delay(1000);
 en = 0;
}
//lcd display charecters function
void lcddisplay(unsigned char *q)
{
 int k;
 for (k = 0; q[k] != '\0'; k++)
 {
 lcddat(q[k]);
 }
 delay(10000);
```

```
}
// assign keypad character value function
char keypad()
{
 int x = 0;
 while (x == 0)
 {
 // assign values for first row
 keyrow1 = 0;
 keyrow2 = 1;
 keyrow3 = 1;
 keyrow4 = 1;
 if (keycolumn1 == 0)
 {
 lcddat('*');
 delay(1000);
 x = 1;
 return '1';
 }
 if (keycolumn2 == 0)
 {
 lcddat('*');
 delay(1000);
 x = 1;
```

```
return '2';
}
if (keycolumn3 == 0)
{
lcddat('*');
delay(1000);
x = 1;
return '3';
}
// assign values for second row
keyrow1 = 1;
keyrow2 = 0;
keyrow3 = 1;
keyrow4 = 1;
if (keycolumn1 == 0)
{
lcddat('*');
delay(1000);
x = 1;
return '4';
}
if (keycolumn2 == 0)
{
lcddat('*');
```

```
delay(1000);
x = 1;
return '5';
}
if (keycolumn3 == 0)
{
lcddat('*');
delay(1000);
x = 1;
return '6';
}
// assign values for third row
keyrow1 = 1;
keyrow2 = 1;
keyrow3 = 0;
keyrow4 = 1;
if (keycolumn1 == 0)
{
lcddat('*');
delay(1000);
x = 1;
return '7';
}
if (keycolumn2 == 0)
```

```
{
lcddat('*');
delay(1000);
x = 1;
return '8';
}
if (keycolumn3 == 0)
{
lcddat('*');
delay(1000);
x = 1;
return '9';
}
// assign values for forth row
keyrow1 = 1;
keyrow2 = 1;
keyrow3 = 1;
keyrow4 = 0;
if (keycolumn1 == 0)
{
lcddat('*');
delay(1000);
x = 1;
return '*';
```

```
 }
 if (keycolumn2 == 0)
 {
 lcddat('*');
 delay(1000);
 x = 1;
 return '0';
 }
 if (keycolumn3 == 0)
 {
 lcddat('*');
 delay(1000);
 x = 1;
 return '#';
 }
 }
}
// password check function and run the door motor
void check()
{
 // compare the input value with the assign password value
 if (pin[0] == Epin[0] && pin[1] == Epin[1] && pin[2] == Epin[2] && pin[3] ==
Epin[3] && pin[4] == Epin[4])
 {
```

```
delay(1000);
lcdcmd(0x01); //decimal value: 1
lcdcmd(0x81); //decimal value: 129
// show pin is correct
lcddisplay("PIN CORRECT");
delay(1000);
// door motor will run
motorpin1 = 1;
motorpin2 = 0;
lcdcmd(0xc1); //decimal value: 193
// show the door is unlocked
lcddisplay("DOOR OPENED");
delay(10000);
motorpin1 = 1;
motorpin2 = 0;
lcdcmd(0x01); //decimal value: 1
}
else
{
lcdcmd(0x01); //decimal value: 1
lcdcmd(0x80); //decimal value: 128
lcddisplay("WRONG PIN");
delay(1000);
lcdcmd(0x01); //decimal value: 1
```

}

}

// end

## Future Scope:

• We can send this data to a remote location using mobile or internet.

• We can add fingerprint sensor so entry will be allowed for the authorized person using their fingerprints.

• We can add fire, wind and LPG sensors so that, the doors will automatically open.