Gas Cylinder Leakage Detection, Weight checking & Automatic Cylinder booking System over IOT

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR, ANANTHAPURAMU

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Y.SIVA KUMARI	**(198R1A0540)**
P.HIMA BINDU	**(198R1A0524)**
G.SREENIVASULU	**(208R5A0501)**
U.LAXMAN SAI KIRAN	**(198R1A0537)**

Under the esteemed guidance of

Dr. G. RAMA SUBBA REEDY, M.E, Ph.D

Professor & Head, Dept. CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAI RAJESWARI INSTITUTE OF TECHNOLOGY

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi, and Affiliated to J.N.T.U.A. Ananthapuramu)

Lingapuram (V), Proddatur, Kadapa (Dist.) -516360

2019-2023

SAI RAJESWARI INSTITUTE OF TECHNOLOGY

AN ISO 9001:2015 CERTIFIED INSTITUTION

(Approved by AICTE, New Delhi and Affiliated to J.N.T.U.A., Ananthapuramu)

Lingapuram(V), Proddatur-516 360, Kadapa (Dist.), A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is to certify that the Project Report entitled

Gas Cylinder Leakage Detection, Weight checking & Automatic Cylinder booking System over IOT

is the bonafide work done and

Submitted by

Y.SIVA KUMARI	**(198R1A0540)**
P.HIMA BINDU	**(198R1A0524)**
G.SREENIVASULU	**(208R5A0501)**
U.LAXMAN SAI KIRAN	**(198R1A0537)**

In the Department of Computer Science and Engineering, Sai Rajeswari Institute of Technology, Proddatur affiliated to J.N.T.U.A., Ananthapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during Academic Year 2021-2022.

Submitted on: _____

Internal Guide	HOD
Dr. G. RAMA SUBBA REDDY, M.E, Ph.D	**Dr. G.RAMA SUBBA REDDY,M.E, Ph.D**
Professor	**Professor**

Internal Examiner	External Examiner

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual effort but the guidance, encouragement, and cooperation of intellectuals, elders, and friends. We would like to take this opportunity to thank them all.

We wholeheartedly express our gratitude and esteemed regards to Project guide **Dr. G. RAMASUBBA REDDY, M.TECH, Ph.D.,professor**, in the Department of Computer Science and Engineering, forproviding us his valuable gratitude and inspiration in carrying out our project studies. His constant support and encouragement enabled us to complete this work successfully.

We express our sincere thanks to Project Co-Ordinator **Mr. M. STANLYWIT, M.E,MBA,** Associate Professor in the Department of Computer Science and Engineering, for providing us his valuable gratitude and inspiration in carrying out our project studies.

With a deep sense of gratitude, we acknowledge **Dr. G. RAMASUBBA REDDY, M.E, Ph.D., professor** Head of the Dept., Computer Science & Engineering, for his valuable support and help in completing our project successfully.

We express our sincere thanks to **Dr. PANDURANGAN RAVI**, M.E, Ph.D., our beloved Principal, for his encouragement and suggestions during our course of study.

We feel honored for placing our warm salutation to **THE MANAGEMENT** Sai Rajeswari Institute of Technology, Proddatur, which allowed us to obtain a strong base in B. Tech and profound knowledge.

Finally, we would like to express our sincere thanks to all the **Faculty Members** of the C.S.E Department, Lab Technicians, Friends & Family members, and all, who have helped us to complete this work successfully.

Y. SIVA KUMARI (198R1A0540)
P. HIMA BINDU (198R1A0524)
G. SREENIVASULU (208R5A0501)
U.LAXMAN SAI KIRAN (198R1A0537)

CONTENTS

LIST OF ABBREVIATIONS

S.NO	Abbreviation	Description
1	GSM	Global System Monitoring
2	LPG	Liquified Petroleum Gas
3	OS	Operating System
4	MVC	Model view controller
5	LCD	Liquid-crystal display
6	SVGA	Standard video graphic array
7	UML	Unified modeling language
8	CRC	Classes, Responsibilities, and collaborators
9	ER	Entity Relationship
10	PC	Personal computer

ABSTRACT

LPG Cylinder is the most commonly used domestic fuel in daily human life. Not only in the hotel, and homes for cooking, but also in several industrial sectors. By this, the demand for LPG usage is increasing rapidly. Meanwhile, this high usage of LPG sometimes leads to gas leakage which may cause a dangerous explosion. There have been many mishaps due to the etonation of LPG cylinders and in some of the occurrences, it is due to the laxity of gas leakage.

To overcome this, our designed system can help for monitor and detect gas leakage. To detect the gas in LPG, the MQ-6 sensor is used, and for any gas leakage that arises the sensor detects by making an alert with a buzzer that can help people, and an alert message is sent to the registered mobile number of the User.

Another major problem faced by the users of LPG Cylinders is the in-opportune fatigue of gas cylinders This proposed system will make the entire LPG cylinder booking course of action automated without human mediation. This apparatus continuously monitors the weight of the cylinder and once it reaches the lightest doorstep value, it will automatically send a message to the permitted LPG Agent so that they can dispatch the LPG cylinder in time. This will eventually help the consumer know when to restore the cylinder.

Keywords: IoT, MQ-6 sensor, Load cell, Buzzer.

1. INTRODUCTION

LPG plays a significant role in our day to day life as there are various ways to use LPG like cooking, vehicle fuel, etc. Gas leakage results in various accidents leading to both material loss and human injuries. The danger of explosion, firing, and suffocation is predicated on their physical properties like toxicity, flammability, etc. The number of deaths because of the explosion of gas cylinders has been increasing in recent years. The LPG or propane is a flammable mixture of hydrocarbon gases used as fuel in many applications like homes, hostels, industries, automobiles, etc. thanks to its desirable properties which include high calorific value, less smoke, less soot, and meager harm to the environment. One of the preventive methods to forestall such accidents is to install a gas leakage detection kit at vulnerable places.

Sometimes the users find it difficult to book a refill because they are not aware of the remaining amount of LPG within the cylinder as they forget the date of installation of the cylinder because of their busy lives and end up booking the cylinder either too early or too late. One of the main drawbacks of the present system is safety.

This project aims to present such a design which will automatically detect gas leakage in vulnerable areas and forestall dangerous situations. The proposed system continuously monitors the amount of the LPG within the cylinder using load cell which is connected to the Arduino UNO R3 which successively displays the amount of the remaining gas through an LCD module and also automatically alert the user to refill the gas, if the gas level reaches below the threshold value it immediately alerts the user about the status of the gas level within the cylinder. The MQ-6 gas sensor is employed to detect any gas leakage, when there is a leakage and the amount of the propane and butane (LPG components) are over the threshold value then it alerts the user about the gas leakage through the buzzer, also it turns off the main power supply of the house, and the main supply of the gas is turned off to prevent explosion and fire accidents. Aside from leakage detection our system also detects smoke and automatically activates the fan and in case of fire accidents, the system detects fire and reports to the fire brigade through an alert message. This paper presents an LPG leakage detection and alert system to avoid fire accidents and to provide safety.

This system is also designed to measure the weight of the cylinder continuously and as soon as it reaches the minimum threshold it will automatically register your LPG Cylinder booking through GSM machinery by sending an SMS to the retailer company and also sending an alert to the user at the same time. This will eventually help the consumer know when to restore the cylinder.

RELATED WORK

To investigate Gas leakage and alarming to alert the citizens about leakage, who are situated locally and remotely located through this system. Examinations by oil organizations found that most LPG customers are ignorant of the security checks of gas cylinders. Another reason is an unauthorized filling of LPG cylinders likewise causes misadventures. There is a need for an agenda to detect and avoid leakage of LPG.

- To detect the leakage of the LPG system.
- To alert the people about the gas leakage by sending messages through text messages, and alarming the Buzzer.
- To alert the gas agency about the exhaustion of gas and to book a new cylinder.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In the existing system, if the LPG gas is going to leak, then the MQ-6 gas sensor will sense the LPG and butane gas concentrates in the air either at home or in industry, it will give a HIGH pulse on its DO pin and Arduino constantly reads its DO pin. When Arduino receives a HIGH pulse from the LPG Gas sensor module it displays that LPG Gas Leakage Alert message on the 16x2 LCD and stimulates a buzzer which beeps again until the gas detector module doesn't recognize the gas in the environment.

Arduino device will produce the buzzer sound based on the time duration by giving instructions in code. When Arduino gets a LOW pulse from the LPG Gas detector module, then LCD will show that No LPG Gas Leakage alert message. Arduino manages the complete process of this system like reading the LPG Gas sensor module output, sending a message to LCD, and stimulating the buzzer. We can set the sensitivity of this sensor module by an inbuilt potentiometer located on it.

2.2 DISADVANTAGES OF EXISTING SYSTEM

- There is no automatic booking system. The user should book the gas manually.
- The user is not able to know how much gas remains.
- User will not hear when far from the buzzer.

2.3 PROPOSED SYSTEM

Proposed System overcomes the lack and inconvenience of existing system. So this proposed system providing functionalities like detecting gas leakage and informing user if there is any leakage occurs. Also subscriber will get the status of gas level of cylinder. The working criteria or principles of any system is mainly dependent on the microcontroller unit which controls the all the functionalities of the devices. In this system the Renesas microcontroller acts as a conditional switch. It performs two set of action depending upon the condition present. It triggers the buzzer and the ALCD to display the message "Gas Leak Detected" when the leakage of the gas is detected by the sensor. The other action is to display the default message on LCD when there is no leakage of gas is detected by the sensor. If the sensor detects the presence the gas in that particular area/place then the GSM module will send "Gas Leak detected" message to the subscriber mobile number. If no gas is detected by the sensor in that particular area/place then the GSM module will not send any messages to the subscriber. Bluetooth Module is included in this device to make the user aware about the leakage of gas taking place at their house in their absence so that user can take necessary actions immediately to prevent a fire accident and can also avoid the losses.

To design LPG leakage detection and Automatic gas booking system for use in Home and Industry. Such a monitoring system will be used to automatically LPG gas booking and Leakage detection. Here we can use the devices like weight stand (load cell) for measuring the gas cylinder level. When gas level reaches below than the threshold value/level which is preloaded in the system. The load cell output is in few mV, so these outputs sends to ADC to convert that value can to digital value and provide to the amplifier through the Renesas controller and automatically booked the gas cylinder using a GSM module using SMS method and display the information on ALCD. MQ-6 gas sensor is used in this proposed system to detect the leakages of gas in the cylinder. When the gas leakage is detected by the sensor and information is sent to the user by SMS (short message service) and immediately controller alerts the customer using GSM module, display particular information on LCD, and it sends alert signal to buzzer device. In proposed System we have two main modules.

A. LPG leakage detection

B. Auto gas booking

In LPG leakage detection is done by gas sensor which is connected with Renesas controller. When gas is detected simultaneously system inform the user about the gas leakage by sending the SMS, beep signal is on the buzzer and also message displaying on LCD about gas leakage. In Proposed System Automatic Gas Booking is accomplish alert system with none human intervention. Our system helps customers to increase their safety and protect their life and properties from fire accidents.

The main objective of our system is to measure the gas present in the cylinder when weight of the cylinder is less than the fixed load by using weight sensor. The gas service provider gets the order message for new cylinder and subscriber received the booking and confirmation messages. Thus the system developed by us will somehow help the LPG user will live their life safely and comfortably.

2.4 ADVANTAGE OF THE PROPOSED SYSTEM

1) It insures the security from the gas leakage and hazards.
2) It is very less time consuming and cylinder replace in time.
3) Easy implementation.
4) It is fully automated system; errors due to human are controlled
5) This system will be sent an alert notification to the user that the gas is going to be completing.
6) The Buzzer alert system is present for the user.
7) Automatically Cylinder Booking system when the gas is completed.
8) Mobile interaction is present.
9) It sends a notification to the user after booking the Gas Cylinder.
10) The user can able to receive the notification that gas has just been booked.
11) Cost-effective.

3. SYSTEM REQUIREMENTS SPECIFICATION

3.1 SOFTWARE REQUIREMENTS

- Domain: IoT
- Operating System: Windows 7 or higher
- Languages: C++
- Arduino IDE: Version 1.8.19

3.2 HARDWARE REQUIREMENTS

- Arduino UNO
- MQ-6 Sensor
- Load cell
- LCD
- Buzzer
- Jumper wires
- Power supply
- GSM module

4. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- OPERATIONAL FEASIBILITY

ECONOMICAL FEASIBILITY

This Economical Feasibility is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget, and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

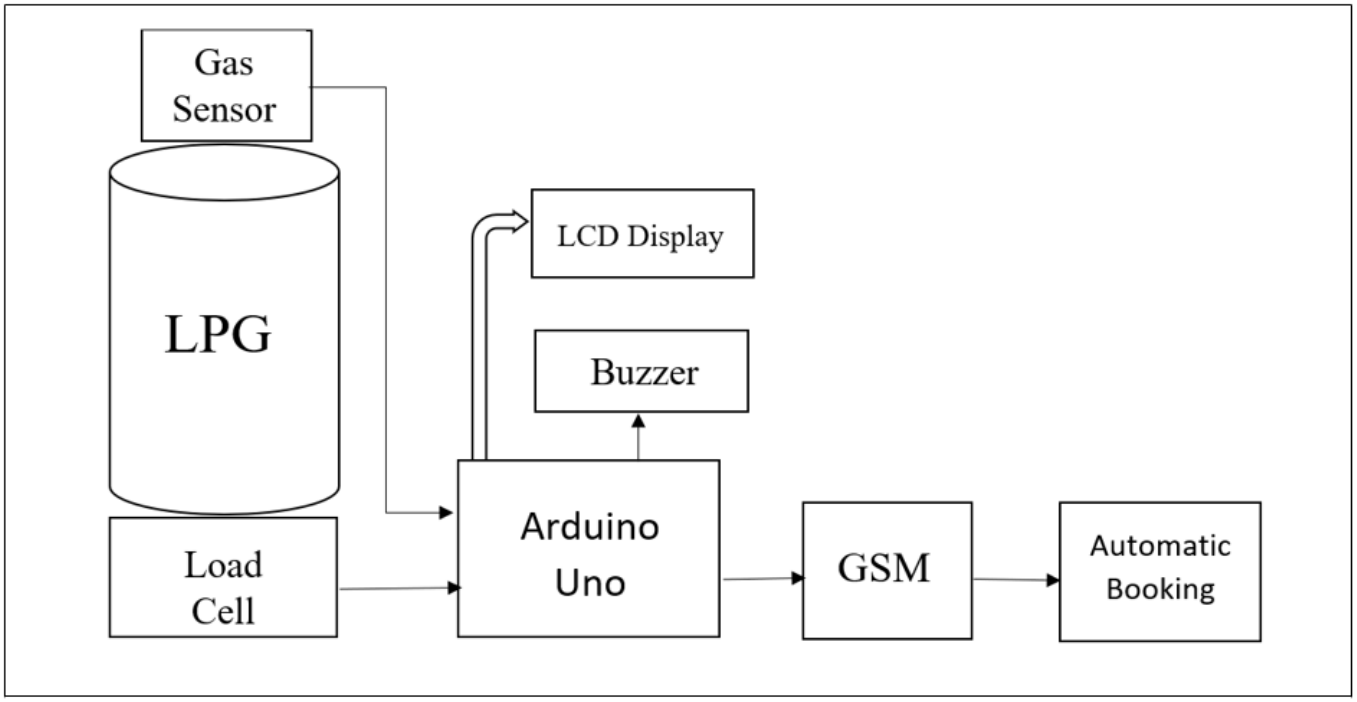
This Technical Feasibility is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

OPERATIONAL FEASIBILITY

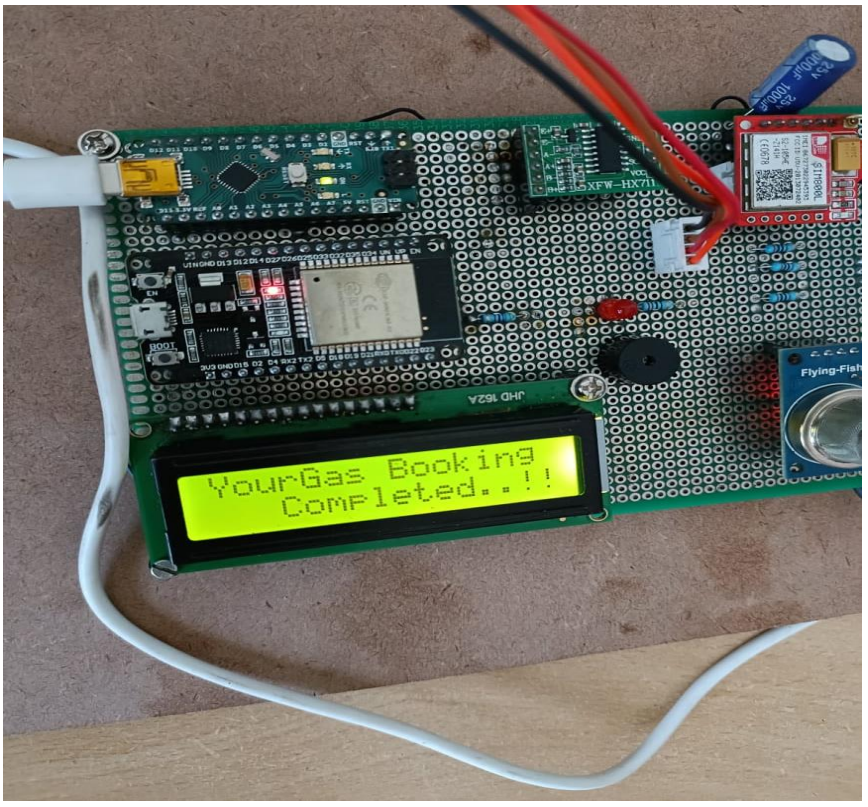
The aspect of Operational Feasibility is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTHURE



5.2 PROJECT PICTURE



5.3 UML APPROACHES

UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form, UML is comprised of two major components:a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing, and documenting the artifacts of the software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the Object Oriented tools in market.
6. Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
7. Integrate best practices.

The Unified Modeling Language is a language form specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with a set of rules and semantics.

1. Class diagram

2. Use case diagram

3. Sequence diagram

4. Activity diagram

5. Collaboration diagram

5.3.1 CLASS DIAGRAM

A class diagram shows a set of classes, interfaces and collaborations and their relationships. A class is an abstraction of real-world items. Class diagrams are the most common diagram found in modeling object-oriented systems. We use class diagrams to illustrate the static design view of a system. Class diagrams that include active classes are used to address the static process view of a system.

Identification of analysis classes

A class is a set of objects that share a common structure and common behavior(the same attributes, operations, relationships, and semantics). A class is an abstraction of real- world items.

There are 4 approaches for identifying classes:

1. Noun phrase approach.

2. Common class pattern approach.

3. Use case Driven Sequence or Collaboration approach.

4. Classes Responsibilities, and collaborators Approach

1. Noun Phrase Approach

The guidelines for identifying the classes:

- Look for nouns and noun phrases in the use cases.
- Some classes are implicit or taken from general knowledge.
- All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.

- Carefully choose and define the class names.

After identifying the classes, we have to eliminate the classes following types of:

- Redundant classes.
- Adjective classes.

2. Common class pattern approach

The following are the patterns for finding the candidate classes:

- Concept class.
- Events class.
- Organization class
- Peoples class
- Places class
- Tangible things and devices class.

3. Use case driven approach

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

4. CRC approach

The process consists of the following steps:

a. Identify classes' responsibilities (and identify the classes)

b. Assign the responsibilities

c. Identify the collaborators

Super-sub class relationships

Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class). This is based on inheritance.

Guidelines for identifying the super-sub relationship, a generalization are

1. **Top-down**: Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub-classes have significant behavior.

2. **Bottom-up:** Look for classes with similar attributes or methods. Group them by moving thecommon attributes and methods to an abstract class. You may have to alter the definitions a bit.

3. **Reusability:** Move the attributes and methods as high as possible in the hierarchy.

4. **Multiple inheritances:** Avoid excessive use of multiple inheritances. One way of getting the benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as attribute.

5. **Aggregation or a-part-of relationship:** It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very difficultly. The major properties of this relationship are transitivity and anti-symmetry.

6. **Assembly:** It is constructed from its parts and an assembly-part situation physically exists.

7. **Container:** A physical whole encompasses but is not constructed from physical parts.

8. **Collection member:** A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by a solid diamond.

5.3.2 USECASE DIAGRAM

A Use Case illustrates a unit of functionality provided by the system. The main purpose of the Use-Case diagram is to help development teams visualize the functional requirements of a system, including the relationship of "actors" to essential processes, as well as the relationships among different use cases. Use Case diagrams generally show groups of Use Cases -- either all use cases for the complete system, or a breakout of a particular group ofuse cases with related functionality. A Use-Case diagram is typically used to communicate the high-level functions of the system and the system's scope. A use case diagram shows a setof use cases and actors and their relationships. We apply use case diagrams to illustrate the static use case view of a system. Use Case diagrams are especially important in organizingand modeling the behavior of a system.

A Use Case in software engineering and systems engineering is a description of a system's behavior as it responds to a request that originates from outside of that system. In other words, a use case describes "who" can do "what" with the system in question. The Use Case technique is used to capture a system's behavioral requirements by detailing scenario-driven threads through the functional requirements. Use cases describe the system from the user's point.

Use Cases describe the interaction between one or more actors (an actor that is the initiator of the interaction may be referred to as the 'primary actor') and the system itself, represented as a sequence of simple steps. Actors are something or someone which exists outside the system ('black box') under study, and that take part in a sequence of activities in a dialogue with the system to achieve some goal. Actors may be end users, other systems, or hardware devices.

ELEMENTS OF A USE CASE DIAGRAM

A use case diagram captures the business processes carried out in the system. Normally, domain experts and business analysts should be involved in writing use cases. Use cases are created when the requirements of a system need to be captured. A use case diagram is quite simple in nature and depicts two types of elements: one representing the business roles and the other representing the business processes.

Actors: An actor portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are the actual business roles of users in a given system.An actor in a use case diagram interacts with a use case. For example, for modeling a banking application, a customer entity represents an actor in the application. Similarly, the person whoprovides service at the counter is also an actor. But it is up to you to consider what actors make an impact on the functionality that you want to model. If an entity does not affect a certain piece of functionality that you are modeling, it makes no sense to represent it as an actor

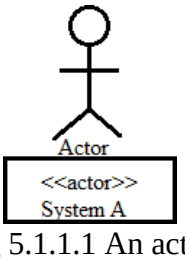


Fig 5.1.1.1 An actor

Use Case: A Use Case in a use case diagram is a visual representation of distinct business functionality in a system. The key term here is "distinct business functionality." To choose a

business process as a likely candidate for modeling as a use case, you need to ensure that the business process is discrete in nature. As the first step in identifying use cases, you should list the discrete business functions in your problem statement. Each of these business functions can be classified as a potential use case. Remember that identifying use cases is a discovery rather than a creation. As business functionality becomes clearer, the underlying use cases become more easily evident.

Fig 5.1.1.2 a use case

System boundary: A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases also need to have definitive limits defined. A system boundary of a use case diagram defines the limits of the system. The system boundary is shown as a rectangle spanning all the use cases in the system. To draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

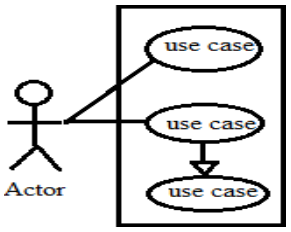


Fig 5.1.1.3 System boundary

RELATIONSHIP IN USE CASES:

Use Cases share different kinds of relationships. A relationship between two use cases is basically a dependency between the two use cases. This reuse of an existing use case using different types of relationships reduces the overall effort required in defining use cases in a system. A similar reuse established using relationships, will be apparent in the other UML diagrams as well.

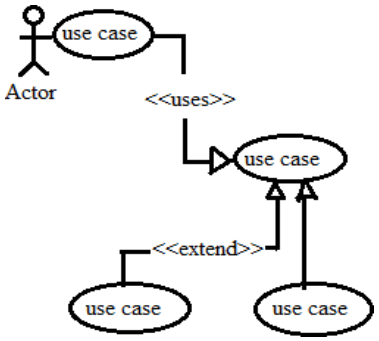


Fig 5.1.1.4 Relationships in use cases

Use case relationships can be one of the following:

- **Include:** When a use case is depicted as using the functionality of another use case in a diagram, this relationship between the use cases is named as an *include* relationship. Literally speaking, in an *include* relationship; a use case includes the functionality described in the use case as a part of its business process flow. An include relationshipis depicted with a directed arrow having a dotted shaft. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow. The stereotype "<<include>>" identifies the relationship as an include relationship.

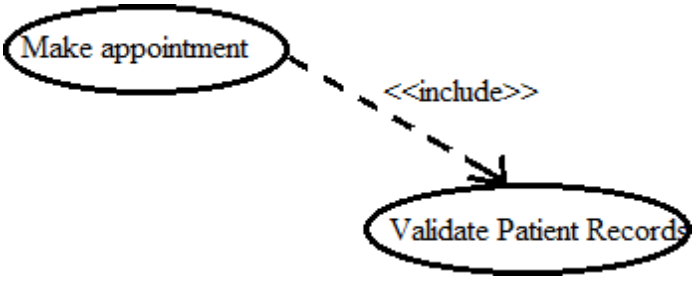
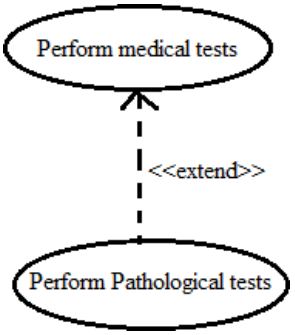


Fig 5.1.1.5 Include relationship

For example, in Figure 6, you can see that the functionality defined by the "Validate patient records" use case is contained within the "Make appointment" use case. Hence,whenever the "Make appointment" use case executes, the business steps defined in the"Validate patient records" use case are also executed.

- **Extend:** In an extend relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case. An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow. The stereotype "<<extend>>" identifies the relationship as an extend relationship.



Extend: It is an example of an extend relationship between the "Perform medical tests" (parent) and "Perform Pathological Tests" (child) use cases. The "Perform Pathological Tests" use case enhances the functionality of the "Perform medical tests" use case. Essentially, the "Perform Pathological Tests" use case is a specialized version of the generic "Perform medical tests" use case.

- **Generalizations:** A generalization relationship is also a parent-child relationshipbetween use cases. The child use case in the generalization relationship has theunderlying business process meaning, but is an enhancement of the parent use case. In a use case diagram, generalization is shown as a directed arrow with a trianglearrowhead. The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.

5.3.3 SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and the messages sent and received by those objects. The objects are typically named or anonymous instance of classes, but may also represent instance of other things, such as collaborations, components and nodes. A sequence diagram is a graphical view of a scenario that shows object interaction in a time-

based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Object: An object has state, behaviour, and identity. The structure and behaviour of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance. The object icon is similar to a class icon except that the name is underlined. An object's concurrency is defined by the concurrency of its class.

Message: A message is the communication carried between two objects that trigger an event. Message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

Link: A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

5.3.4 COLLABORATION DIAGRAM

Collaboration diagrams and Sequence diagrams are alternate representations of an**I**nteraction. A Collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows objectinteraction in a time-based sequence. Collaboration diagrams show objects, their links, andtheir messages. They can also contain simple class instances and class utility instances. Each Collaboration diagram provides a view of the interactions or structural relationships thatoccur between objects and object-like entities in the current model. These diagrams are usedto indicate the semantics of the primary and secondary interactions. They also show thesemantics of mechanisms in the logical design of the system.

Message icons: A message icon represents the communication between objects indicating that an action will follow. The message icon is a horizontal, solid arrow connecting two lifelines together. A message icon can appear in 3 ways: message icon only, message icon with sequence number, and message icon with sequence number and message label.

There are two types of numbering schemes.

1. Flat numbered sequence: In this message are numbered as 1, 2, 3….

2. Decimal numbered sequence: In this the messages are given numbers as 1.1, 1.2, 1.3……It makes clear which operation iscalling which other operation.

DIFFERENCES BETWEEN SEQUENCE AND COLLABORATION DIAGRAMS:

- Sequence diagram is easy to read.
- Collaboration diagram can be used to indicate how objects are statically connected.
- There is no numbering in sequence diagram.
- Sequence diagram shows the links between objects in a time-based sequence.

5.3.5 STATE DIAGRAM

State chart diagram is used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So, State chart diagrams are useful to model reactive systems. Reactive systems can be defined as a system that responds to external or internal events. State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So, the most important purpose of State chart diagram is to model life time of an object from creation to termination**.**

State chart diagrams are also used for forward and reverse engineering of a system. But the main purpose is to model reactive system.

Following are the main purposes of using State chart diagrams:

1. To model dynamic aspect of a system.

2. To model life time of a reactive system.

3. To describe different states of an object during its life time.

4. Define a state machine to model states of an object.

Contents

Simply state and composite states Transitions, including events and actions

Common use

They are used to model the dynamic aspects of a system.

5.3.1 ACTIVITY DIAGRAM

An activity diagram shows the flow from activity to activity within a system. An activity shows a set of activities, the sequential or branching flow from activity to activity and objects that act and are acted. We use activity diagrams to illustrate the dynamic view of a system. Activity diagrams emphasize the flow of control among objects. Activity diagrams provide a way to model the workflow of a business process, code-specific information such as a class operation. The transitions are implicitly triggered by completion of the actions in the source activities. The main difference between activity diagrams and state charts is activity diagrams are activity centric, while state charts are state centric.

An activity diagram is typically used for modelling the sequence of activities in a process. An Activity represents the performance of task or duty in a workflow. It may also represent the execution of a statement in a procedure. You can share activities between state machines. However, transitions cannot be shared. An action is described as a "task" that takes place while inside a state or activity.

Actions on activities can occur at one of four times:

- **On entry**: The "task" must be performed when the object enters the state or activity.
- **On exit:** The "task" must be performed when the object exits the state or activity.
- **Do**: The "task" must be performed while in the state or activity and must continue until exiting the state.
- **On event:** The "task" triggers an action only if a specific event is received.
- An **end state** represents a final or terminal state on an activity diagram or state chart diagram.
- A **start state** (also called an "initial state") explicitly shows the beginning of aworkflow on an activity diagram.
- **Swim lanes** can represent organizational units or roles within a business model. Theyare very similar to an object. They are used to determine which unit is responsible forcarrying out the specific activity. They show ownership or responsibility. Transitions cross swim lanes.
- **Synchronizations** enable you to see a simultaneous workflow in an activity diagram Synchronizations visually define forks and joins representing parallel.

5.3.2 COMPONENT DIAGRAM

Component diagrams are used to model physical aspects of a system. Now thequestion is what are these physical aspects, Physical aspects are the elements like executables, libraries, files, documents etc. which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. Thesediagrams are also used to make executable systems.

Purpose

Component diagrams can be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagramsis used to represent the whole.

Before drawing a component diagram, the following artifacts are to be identified clearly:
Files used in the system. Libraries and other artifacts relevant to the application.
Relationship among the artifacts.

Now after identifying the artifacts the following points needs to be followed:
Use a meaningful name to identify the component for which the diagram is to be drawn. Prepare a mental layout before producing using tools. Use notes for clarifying important points.

Now the usage of component diagrams can be described as:

1. Model the components of a system.

2. Model database schema.

3. Model executables of an application.

4. Model system's source code.

Contents

Components, Interfaces, Relationships

5.3.3 DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are usedto describe the static deployment view of a system. Purpose: The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related. Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components. So, most of the

UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topologyof a system. Deployment diagrams are used by the system engineers. The purpose of deployment diagrams can be described as: Visualize hardware topology of a system. Describe the hardware components used to deploy software components. Describe runtime processingnodes.

5.4 UML DIAGRAMS

5.4.1 CLASS DIAGRAM

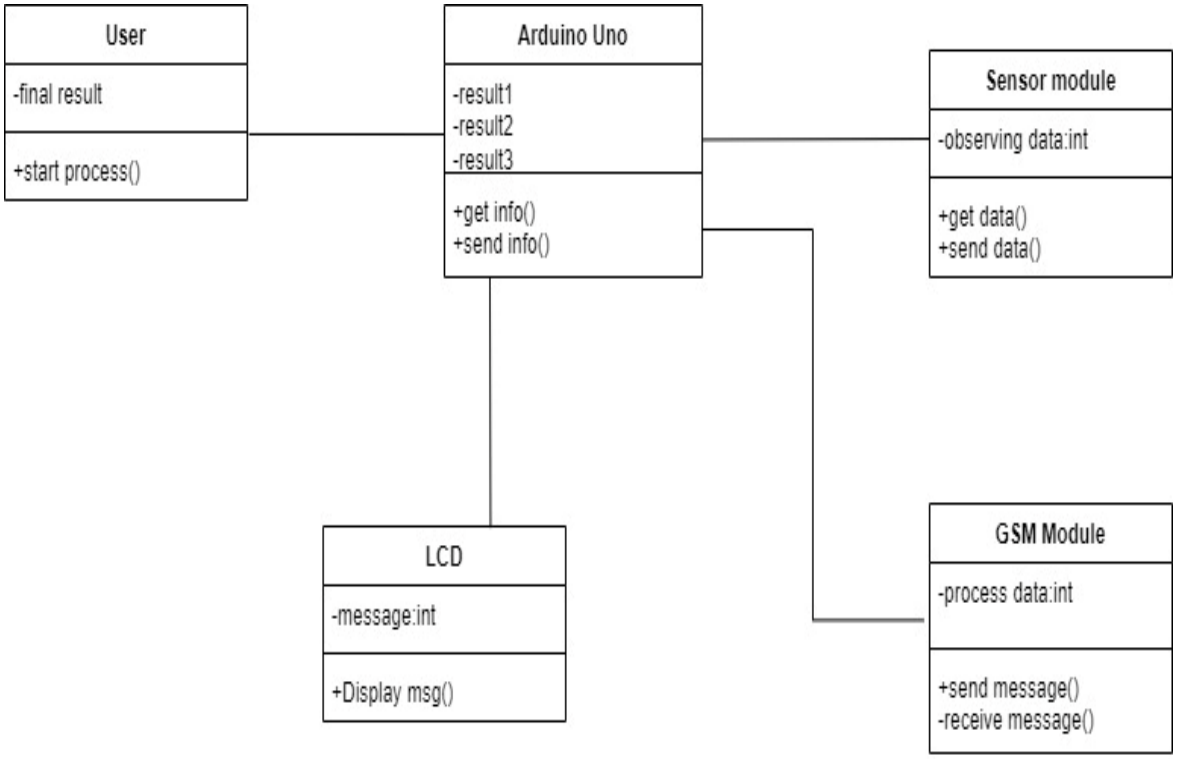


Figure 5.3.1: Class Diagram

5.4.2 Use case Diagram

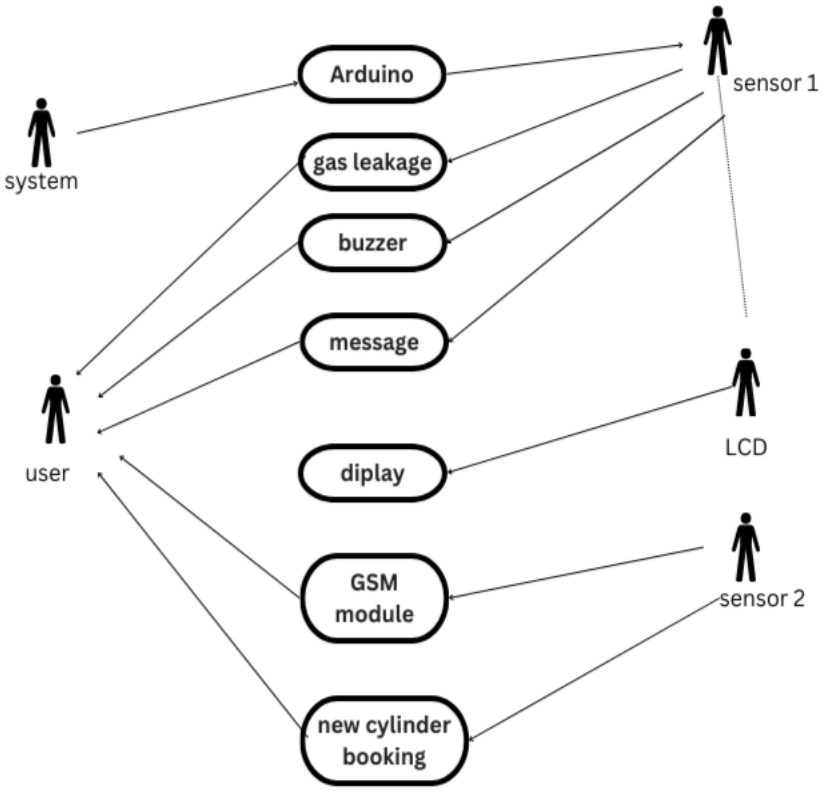


Figure 5.3.2: Use case Diagram

5.4.3 Sequence Diagram

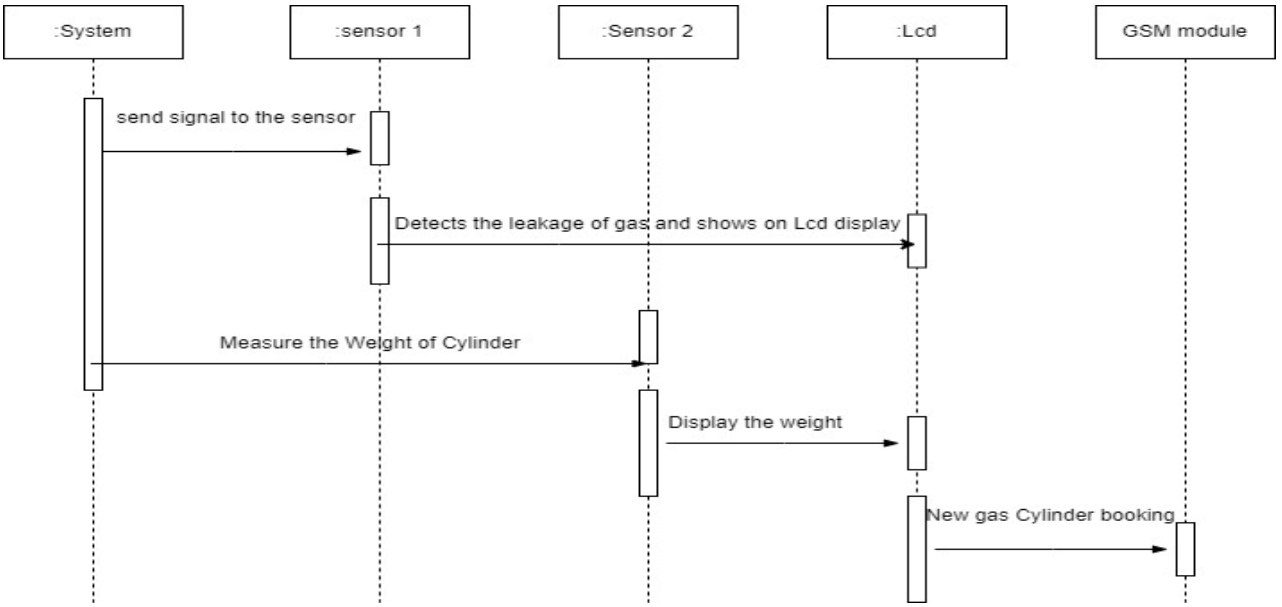


Fig 5.3.3: Sequence Diagram

5.4.4 State Diagram

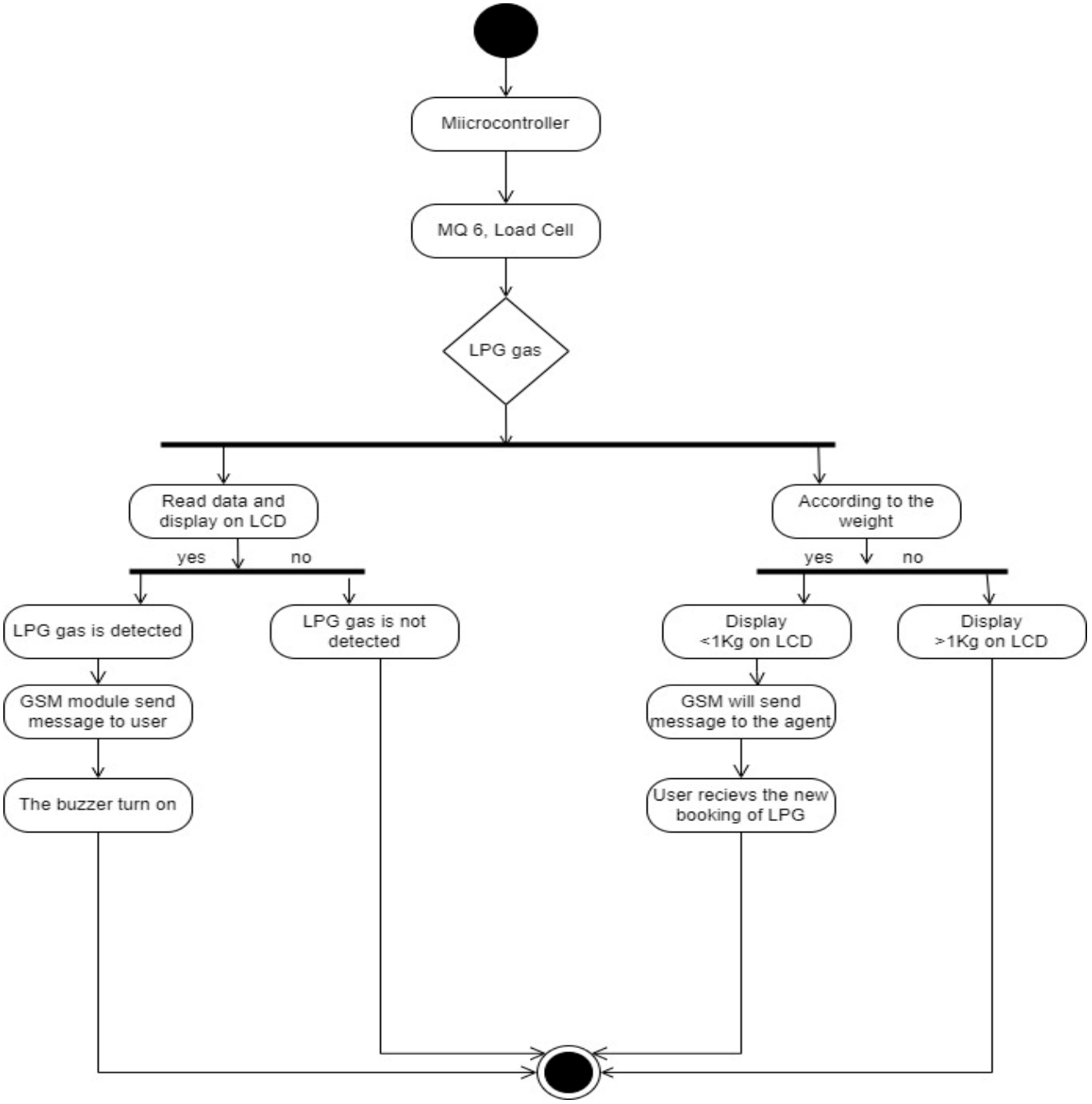


Fig 5.3.4 State Diagram

5.4.5 Activity Diagram

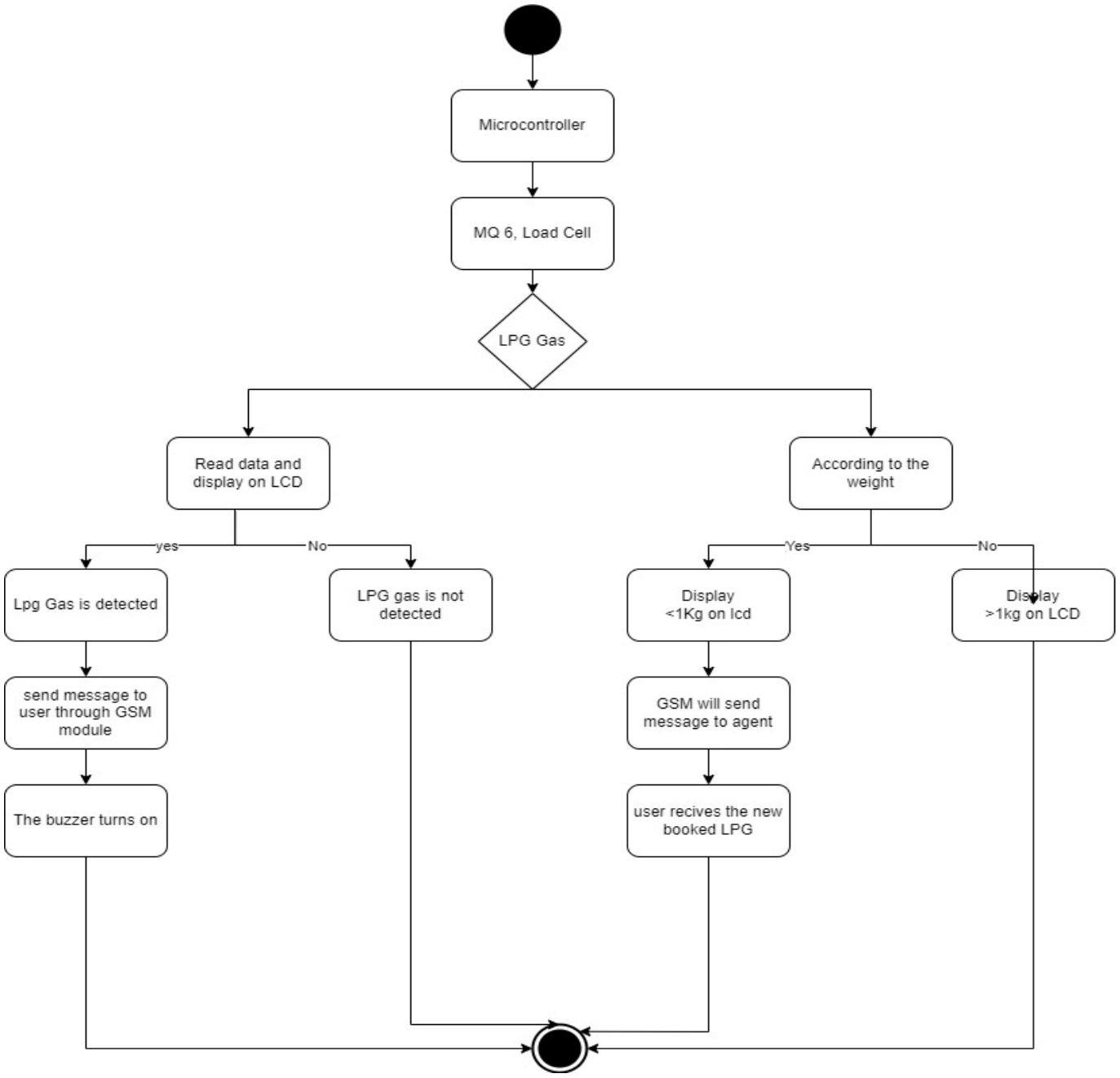


Figure 5.3.5: Activity Diagram

5.4.6 Component Diagram

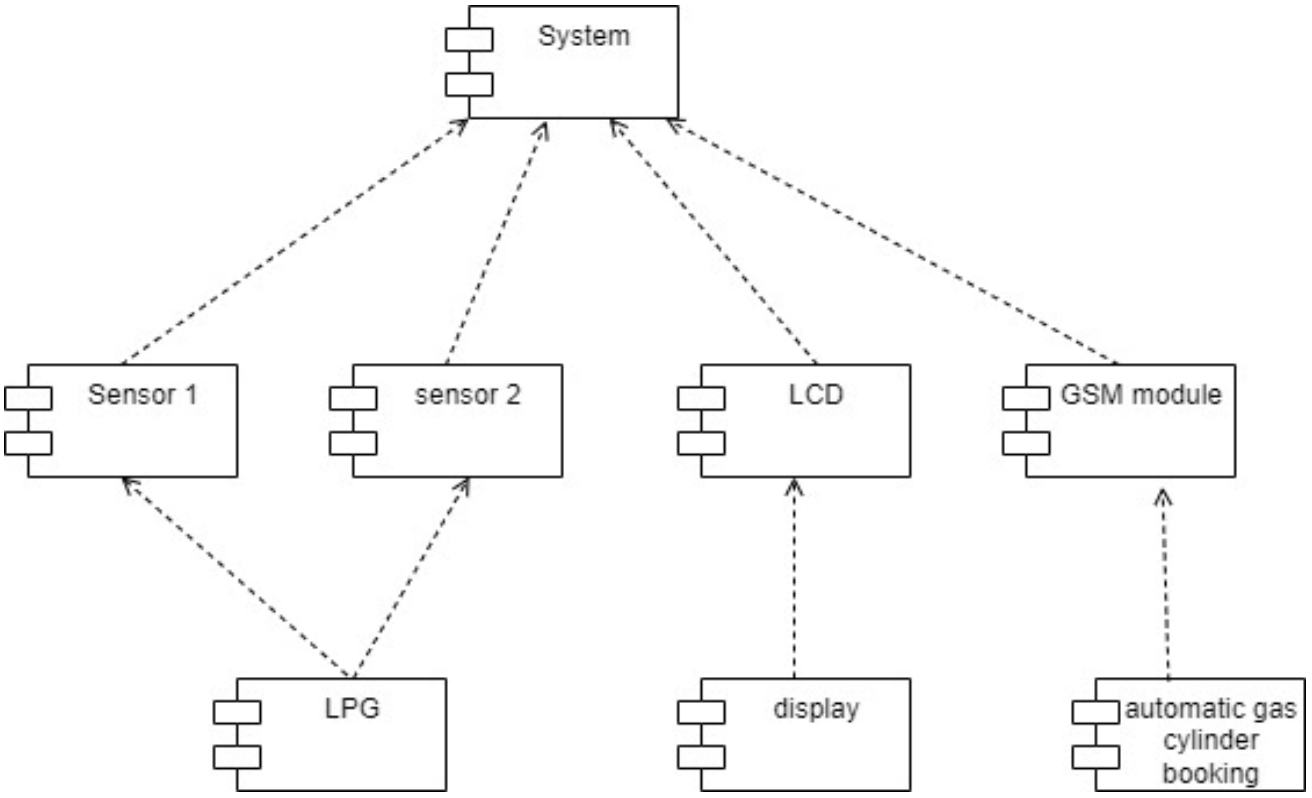


Fig 5.3.6 Component Diagram

5.4.7 Deployment Diagram

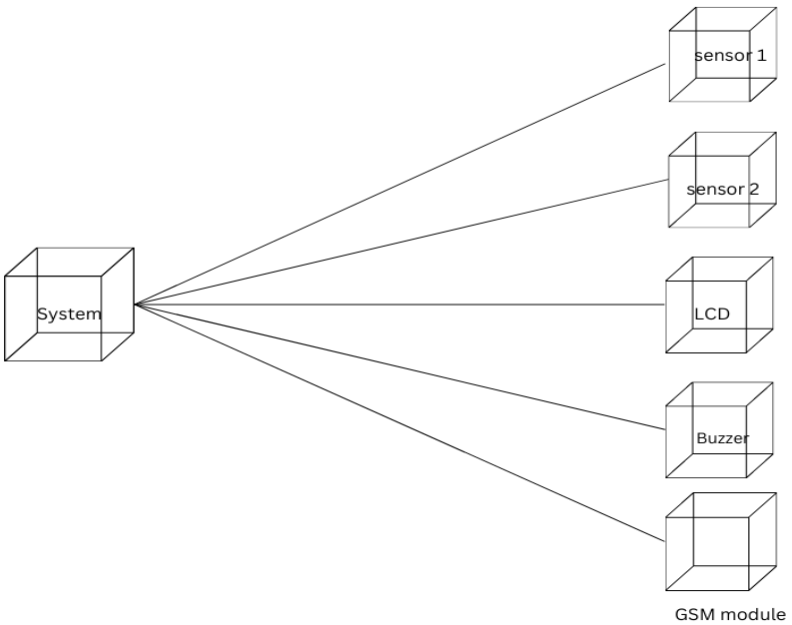


Fig 5.3.7 Deployment Diagram

6. SYSTEM CODING

6.1 Sample Code

```
#define BLYNK_TEMPLATE_ID "TMPLPzjtrmF3"
#define BLYNK_TEMPLATE_NAME "GasLeackage"
#define BLYNK_AUTH_TOKEN "YWLGYPZ8wa4vH8kC-6vHDDpeM5vG5PnJ"

/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "kumari";
char pass[] = "123456789";
int SensorPin = 13;

WidgetLCD lcd(V0);

void  SensorData()
{
  int SensorValue = digitalRead(SensorPin);
  Serial.println(SensorValue);
  if  (SensorValue == 0)
    {
      digitalWrite(14, HIGH);
      delay(1000);
      lcd.clear();
      lcd.print(0, 0, "Gas Detected");
      Blynk.logEvent("system_1","Gas Detected");
      }
  else
  {
    lcd.clear();
    digitalWrite(14, LOW);
  }

}

void setup()
{
  // Debug console
  Serial.begin(115200);
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
  pinMode(SensorPin,INPUT);
  pinMode(14, OUTPUT);
}
```

```
void loop()
{
  Blynk.run();
  SensorData();
  delay(1000);
}
```

Code for Booking

```
  hj#include "HX711.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>


LiquidCrystal_I2C Display(0x27,16,2);
SoftwareSerial mySerial(6,5);
#define DEBUG_HX711
#define calibration_factor -7050.0

HX711 scale;

#define LOADCELL_DOUT_PIN  8
#define LOADCELL_SCK_PIN  9
int SensorPin = 13;
float Value=0;
String incomingData;
String message = "";

void LoadCell_setup()
  {
    //Serial.begin(9600);
    Serial.println("HX711 scale demo");
    scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
    scale.set_scale(calibration_factor); //This value is obtained by using the
SparkFun_HX711_Calibration sketch
    scale.tare(); //Assuming there is no weight on the scale at start up, reset the scale to 0
    Serial.println("Readings:");
  }
void LoadCellData()
  {
    Serial.print("Reading: ");
     float Values =((scale.get_units()*100)-100); //scale.get_units() returns a float
    //float Values =values;
    Serial.print(Values);
    Serial.print("Grams"); //You can change this to kg but you'll need to refactor the
calibration_factor
    Serial.println();
    Display.clear();
    Display.setCursor(0,0);
    Display.print("Your Weight");
    Display.setCursor(0,1);
    Display.print(Values);
```

```
    Display.setCursor(8,1);
    Display.print("Grams");
    delay(1000);

    if(Values <=2000 && Values>=150)
    {
      Display.setCursor(0,0);
      Display.print("YourGas Below5KG");
      Display.setCursor(0,1);
      Display.print("PleaseBookYourGas");
      delay(3000);
      Send_setup();
    }
   }

void Send_setup()
  {

    mySerial.begin(9600);
    Serial.println("Initializing...");
    delay(1000);

    mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
    updateSerial();

    mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
    updateSerial();

    mySerial.println("AT+CMGS=\"+918374304499\"");//change ZZ with country code and
xxxxxxxxxxx with phone number to sms
    updateSerial();

    mySerial.print("Booking"); //text content
    updateSerial();
    mySerial.write(26);
}
void Send_setup2()
  {

    mySerial.begin(9600);
    Serial.println("Initializing...");
    delay(1000);

    mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
    updateSerial();

    mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
    updateSerial();

    mySerial.println("AT+CMGS=\"+918374304499\"");//change ZZ with country code and
xxxxxxxxxxx with phone number to sms
    updateSerial();
```

```
    mySerial.print("Booking Process Successfully Completed..!!!"); //text content
    updateSerial();
    mySerial.write(26);
}

void Read_setup()
{
  //Serial.begin(115200);
  mySerial.begin(9600); // baudrate for GSM shield
  mySerial.print("AT+CMGF=1\r");
  delay(100);
  mySerial.print("AT+CNMI=2,2,0,0,0\r");
  delay(100);
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
    {
      mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
    }
   while(mySerial.available())
    {
      Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
    }
}
void receive_message()
{
  if (mySerial.available() > 0)
  {
    incomingData = mySerial.readString(); // Get the data from the serial port.
    Serial.print(incomingData);
    delay(1000);
  }
}

void setup()
{
  Serial.begin(115200);
  Display.init();
  Display.backlight();
  Display.setCursor(4,0);
  Display.print("Welcome To");
  Display.setCursor(0,1);
  Display.print("GasLeackageSystem");
  delay(2000);
  pinMode(SensorPin,INPUT);
  LoadCell_setup();
  Read_setup();
}

void loop()
```

```
{

  receive_message();
  if(incomingData.indexOf("Book")>=0)
  {
    Display.clear();
    Display.setCursor(0,0);
    Display.print("YourGas Booking");
    Display.setCursor(3,1);
    Display.print("Completed..!!");
    Send_setup2();
    delay(5000);
    Display.clear();
    incomingData="";

  }
  LoadCellData();
}
```

6.2. About programming Language

Every full C++ program begins inside a function called "main". A function is simply a collection of commands that do "something". The main function is always called when the program first executes. From main, we can call other functions, whether they be written by us or by others or use built-in language features. To access the standard functions that comes with your compiler, you need to include a header with the #include directive. The next important line is int main(). This line tells the compiler that there is a function named main, and that the function returns an integer, hence int. The "curly braces" ({ and}) signal.

The cout function is the standard C++ way of displaying output on the screen. The quotes tell the compiler that you want to output the literal string as-is (almost). The '\n' sequence is actually treated as a single character that stands for a newline (we'll talk about this later in more detail); for the time being, just remember that there are a few sequences that, when they

C++ is a general-purpose programming language that was developed as an enhancement of The C-language to include object-oriented paradigm. It is an imperative and a **compiled** language.

1. C++ is a high-level, general-purpose programming language designed for system and application programming. It was developed by Bjarne Stroustrup at Bell Labs in 1983 as an extension of the C programming language. C++ is an object-oriented, multi-paradigm language that supports procedural, functional, and generic programming styles.

2. One of the key features of C++ is its ability to support low-level, system-level programming, making it suitable for developing operating systems, device drivers, and other system software. At the same time, C++ also provides a rich set of libraries and features for high-level application programming, making it a popular choice for developing desktop applications, video games, and other complex applications.
3. C++ has a large, active community of developers and users, and a wealth of resources and tools available for learning and using the language. Some of the key features of C++ include:
4. **Object-Oriented Programming:** C++ supports object-oriented programming, allowing developers to create classes and objects and to define methods and properties for these objects.
5. **Templates:** C++ templates allow developers to write generic code that can work with any data type, making it easier to write reusable and flexible code.
6. **Standard Template Library (STL):** The STL provides a wide range of containers and algorithms for working with data, making it easier to write efficient and effective code.
7. **Exception Handling:** C++ provides robust exception handling capabilities, making it easier to write code that can handle errors and unexpected situations.
8. Overall, C++ is a powerful and versatile programming language that is widely used for a range of applications and is well-suited for both low-level system programming and high-level application development.
9. As in C, C++ supports four types of memory management: static storage duration objects, thread storage duration objects, automatic storage duration objects, and dynamic storage duration objects. Static storage duration objects are created before `main()` is entered (see exceptions below) and destroyed in reverse order of creation after `main()` exits. The exact order of creation is not specified by the standard (though there are some rules defined below) to allow implementations some freedom in how to organize their implementation. More formally, objects of this type have a lifespan that "shall last for the duration of the program".
10. C++ is a high-level, general-purpose programming language designed for system and application programming. It was developed by Bjarne Stroustrup at Bell Labs in 1983 as an extension of the C programming language. C++ is an object-oriented, multi-paradigm language that supports procedural, functional, and generic programming styles.
11. One of the key features of C++ is its ability to support low-level, system-level programming, making it suitable for developing operating systems, device drivers, and other system software. At the same time, C++ also provides a rich set of libraries and features for high-level application programming,

12. C++ has a large, active community of developers and users, and a wealth of resources and tools available for learning and using the language. Some of the key features of C++ include:
13. **Object-Oriented Programming:** C++ supports object-oriented programming, allowing developers to create classes and objects and to define methods and properties for these objects.
14. **Templates:** C++ templates allow developers to write generic code that can work with any data type, making it easier to write reusable and flexible code.
15. **Standard Template Library (STL):** The STL provides a wide range of containers and algorithms for working with data, making it easier to write efficient and effective code.
16. **Exception Handling:** C++ provides robust exception handling capabilities, making it easier to write code that can handle errors and unexpected situations.
17. Overall, C++ is a powerful and versatile programming language that is widely used for a range of applications and is well-suited for both low-level system programming and high-level application development.
18. As in C, C++ supports four types of memory management: static storage duration objects, thread storage duration objects, automatic storage duration objects, and dynamic storage duration objects. Static storage duration objects are created before `main()` is entered (see exceptions below) and destroyed in reverse order of creation after `main()` exits. The exact order of creation is not specified by the standard (though there are some rules defined below) to allow implementations some freedom in how to organize their implementation. More formally, objects of this type have a lifespan that "shall last for the duration of the program".

Using Variables

A variable of type char stores a single character, variables of type int store integers (numbers without decimal places), and variables of type float store numbers with decimal places. Each of these variable types - char, int, and float - is each a keyword that you use when you declare a variable. Some variables also use more of the computer's memory to storevalues. It may seem strange to have multiple variable types when it seems like some variable types are redundant. But using the right variable size can be important for making your program efficient because some variables require more memory than others. For now, suffice it to say that the different variable types will almost all be used.

Reading input

Using variables in C for input or output can be a bit of a hassle at first, but bear with it and it will make sense. We'll be using the scanf function to read in a value and then printf to

read it back out. Let's look at the program and then pick apart exactly what's going on. You can even compile this and run it if it helps you follow along.

Basics of Using Functions

Comments in code can be useful for a variety of purposes. They provide the easiest way to set off specific parts of code (and their purpose); as well as providing a visual ”split” between various parts of your code. Having good comments Functions are a big part of programming. A function is a special kind of block that performs a well-defined task. If a function is well-designed, it can enable a programmer to perform a task without knowing anything about how the function works. The act of requesting a function to perform its task is called a function call. Many functions require a caller to hand it certain pieces of data needed to perform its task; these are called arguments. Many functions also return a value to the caller when they're finished; this is called a return value (the return value in the above program is 0). The things you need to know before calling a function are:

- The data type (discussed later) of the arguments and what they mean
- The data type of the return value and what it means many functions use the return value for the result of a computation. Some functions use the return value to indicate whether they successfully completed their work.

Comments

code will make it much easier to remember what specific parts of your code do. Comments in modern flavors of C (and many other languages) can come in two forms: 1 //Single Line Comments (added by C99 standard, famously known as c++ style of 2 comments) and 1 /*Multi-Line 2 Comments 3 (only form of comments supported by C89 standard)*/ Note that Single line comments are a more recent addition to C, so some compilers may not support them. A recent version of GCC5 will have no problems supporting them. This section is going to focus on the various uses of each form of commentary

1. Single-line Comments

What is immediately obvious from reading the code does not belong in a comment. Single-line comments are most useful for simple 'side' notes that explain what certain parts the code do. The best places to put these comments are next to variable declarations, and next to pieces of code that may need explanation. Comments should make clear the intention and ideas behind the corresponding code

2. Multi-line Comments

Multi-line comments are most useful for long explanations of code. They can be used as copyright/licensing notices, and they can also be used to explain the purpose of a block of code. This can be useful for two reasons: They make your functions easier to understand, and they make it easier to spot errors in code. If you know what a block is supposed to do, then it is much easier to find the piece of code that is responsible if an error occurs.

The Four Basic Data Types

In Standard C there are four basic data types. They are int, char, float, and double. We will briefly describe them here, then go into more detail in C Programming/Types5.

- The int type stores integers in the form of ”whole numbers”. An integer is typically the size of one machine word, which on most modern home PCs is 32 bits (4 octets).
- The char type is capable of holding any member of the execution charact-er set6. Itstores the same kind of data as an int (i.e. integers), but typical-ly has a size of one byte.
- float is short for floating point. It stores inexact representations of real numbers, both integer and non-integer values. It can be used with num-bers that are much greater than the greatest possible int.
- The double and float types are very similar. The float type allows you to store single precision floating point numbers

7. IMPLEMENTATION

MODULES

1. GSM module
2. Sensors module
3. Arduino module
4. Power supply module
5. Display module

7.1 GSM MODULE

The Global System for Mobile Communications (GSM) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile devices such as mobile phones and tablets. GSM is also a trade mark owned by the GSM Association.

GSM was intended to be a secure wireless system. It has considered the user authentication using a pre-shared key and challenge–response, and over-the-air encryption. However, GSM is vulnerable to different types of attack, each of them aimed at a different part of the network.

GSM uses General Packet Radio Service (GPRS) for data transmissions like browsing the web. The most commonly deployed GPRS ciphers were publicly broken in 2011.The researchers revealed flaws in the commonly used GEA/1 and GEA/2 (standing for GPRS Encryption Algorithms 1 and
Beginning in the late 2010s, various carriers worldwide started to shut down their GSM networks. Nevertheless, as a result of the network's widespread use, the acronym "GSM" is still used as a generic term for the plethora of <n>G mobile phone technologies evolved from it. GSM digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its own time slot. It operates at either the 900 megahertz (MHz) or 1,800 MHz frequency band. Communication Network: It is used for data security and data transmission. The GSM technology is used which uses mobile stations, base substations, and network systems.

7.2 SENSOR MODULE:

Sensor module consists of the following:

a. MQ-6 Sensor

b. Load cell

a. MQ-6 Sensor:

MQ6 Gas sensor is a Metal Oxide Semiconductor (MOS) type Gas Sensor mainly used to detect the LPG and Butane gas concentration in the air either at home or in industry. This sensor contains a sensing element, mainly aluminum-oxide based ceramic, coated with Tin dioxide, enclosed in a stainless-steel mesh.

Pin Description:

- The VDD power supply 5V DC
- GND used to connect the module to system ground
- DIGITAL OUT, you can also use this sensor to get digital output from this pin, by setting a threshold value using the potentiometer
- ANALOG OUT, this pin outputs 0–5V analog voltage based on the intensity of the gas.

b. Load cell Sensor:

A load cell converts a force such as tension, compression, pressure, or torque into an electrical signal that can be measured and standardized. It is a force transducer. As the force applied to the load cell increases, the electrical signal changes proportionally. The most common types of load cell are pneumatic, hydraulic, and strain gauges.

7.3 ARDUNIO MODULE:

The Arduino microcontroller is an easy-to-use yet powerful single-board computer that has gained considerable traction in the hobby and professional market. The Arduino is open- source which means the hardware is reasonably priced and the development software is free.

This guide is for students in ME 2022, or students anywhere who are confronting the Arduino for the first time. For advanced Arduino users, prowl the web; there are lots of resources.

This is what the Arduino board looks like.

The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a few commands are needed to perform useful functions.

Arduino Hardware:

The power of the Arduino is not its ability to crunch code, but rather its ability to interact with the outside world through its input-output (I/O) pins. The Arduino has 14 digital I/Opins labeled 0 to 13 that can be used to turn motors and lights on and off and read the state of switches.

Each digital pin can sink or source about 40 mA of current. This is more than adequate for interfacing to most devices but does mean that interface circuits are needed to control devices other than simple LEDs. In other words, you cannot run a motor directly using the current available from an Arduino pin, but rather must have the pin drive an interface circuit that in turn drives the motor. To interact with the outside world, the program sets digitalpins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin.

Pin Descriptions:

VCC: Digital supply voltage.

GND: Ground.

Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2: Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability.

Port C (PC5:0): Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low willsource current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/RESET: If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input.

Port D (PD7:0): Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability.

AVCC: AVCC is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter. Note that PC6..4 use digital supply voltage, VCC.

AREF: AREF is the analog reference pin for the A/D Converter

ADC7:6 (TQFP and QFN/MLF Package Only): In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

7.4 POWER SUPPLY MODULE

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a

- **VIN**: The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). Youcan supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **5V:** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

- **3V3**: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **GND:** Ground pins.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfused that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

POWER SUPPLY:

Control supplies, occasionally called control connectors, or just connectors, are accessible in different voltages, with changing current purposes of repression, which is quite recently the most phenomenal most extreme of a compel supply to pass on current to a stack. Subsequently, on the off chance that you create one yourself, you will always know how to repair it, as you will know effectively what area/some part of the circuit is doing what. Also, further, knowing how to make one will permit you to repair the ones you have inception at now got, without squandering your cash on another.

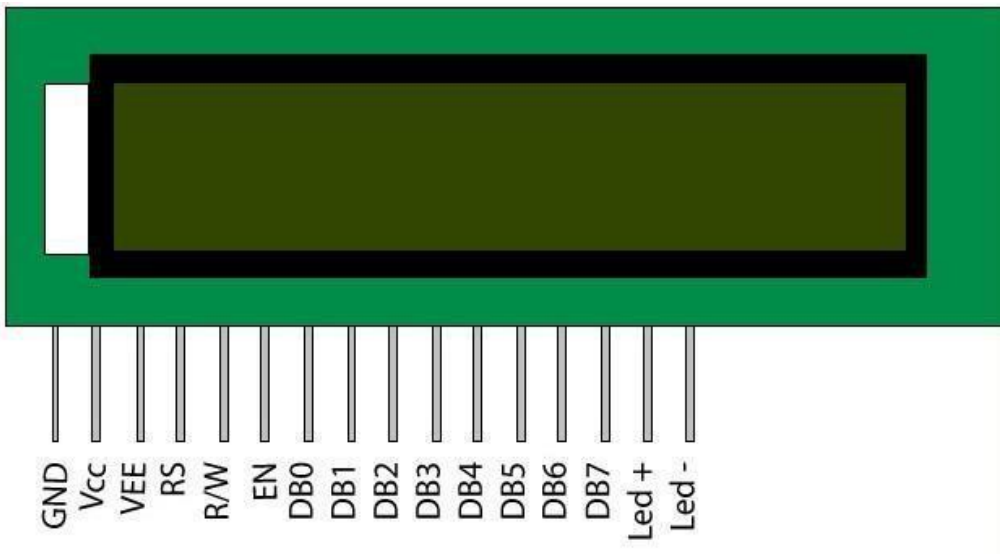
.5 Display module

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs.

The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Clickto learn more about internal structure of a LCD.

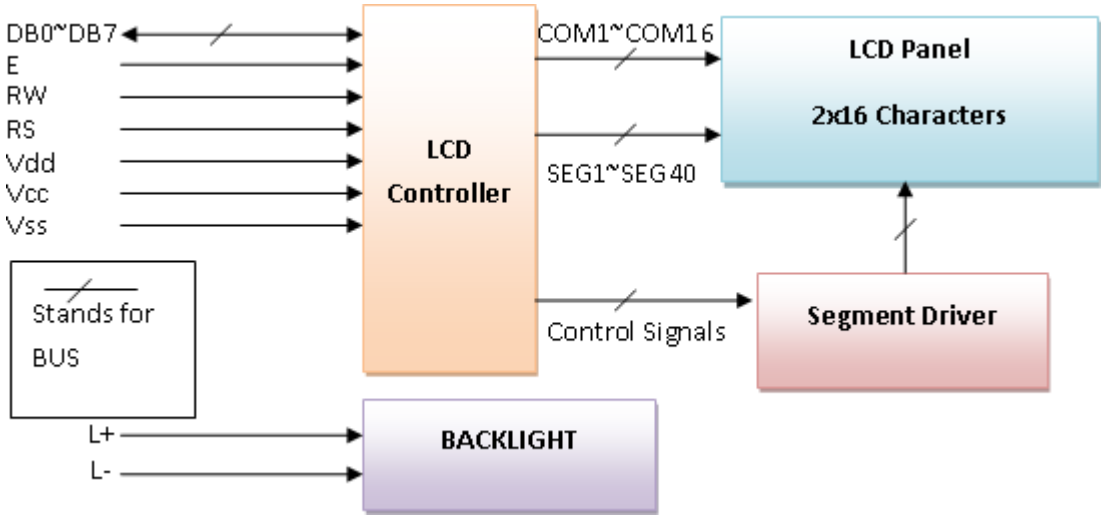
Pin Diagram:



Pin Description:

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc
3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/ write
6	Sends data to data pins when a high to low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

Block Diagram of LCD Display:



BIT MODE

There is lot of stuff that can be done with the LCDs, to start with we will simple display a couple of strings on the 2 lines of the LCD as shown in the image.

Schematic description

- **Data Lines:** In this mode, all of the 8 data lines DB0 to DB7 are connected from the microcontroller to a LCD module as shown the schematic.
- **Control Lines:**' The RS, RW and E are control lines, as discussed earlier.
- **Power & contrast**: Apart from that the LCD should be powered with 5V between **PIN 2(VCC)** and **PIN 1(gnd)**. **PIN 3** is the contrast pin and is output of center terminal of potentiometer (voltage divider) which varies voltage between 0 to 5v to vary the contrast.
- **Back-light:** The PIN 15 and 16 are used as backlight. The led backlight can be powered through a simple current limiting resistor as we do with normal leds.

4 BIT MODES:

Schematic

There are following differences in 4-bit mode.

- Only data lines D4 to D7 are used as shown in the schematic below.
- In code, we need to send the command to select 4-bit mode as shown in the instruction set above.

The main program remains exactly as in 8- b i t mode, we simply include the lcd_4_bit.cto work in 4-bit mode.

8. TESTING

Testing strategies and Validation testing

A Strategy for software testing integrates software test cases into a series of well-planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers the set of activities that ensure that the software that has been built is traceable to customer's requirements.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are testedto uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

Integration Testing

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

Top-down Integration

Top-down integrations is an incremental approach for construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

Bottom-up Integration

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottomup manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

Validation Testing

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocation functions. The extent of testing a system is controlled by many factors, such as the risk involved, limitations of resources, and deadlines. In light of these issues, we must employ a testing strategy that does the "Best" work of finding errors and failures in a product within the given constraints. There are many testing strategies, but testing uses.

Two types of testing:

1. Black Box Testing
2. White Box Testing

Black Box Testing

The concept of black box testing is used to represent a system whose inside working structure is unknown for inspection. In a black box testing, the test is treated as "Black", since its logic is not known is what comes in and what goes out i.e, inputs and outputs are unknown .In black box testing we try various inputs and examine the resulting outputs. We can learn what the box does but nothing about how this conversion is implemented. Blackbox testing woks very nicely on testing objects in an object oriented environment. The black box technique can also be used for scenario based test where the system' s inside may not be available for inspection but the input and output are defined through use test cases or other analysis methods. The black box is an imaginary box that hides its internal working.

White Box Testing

White box testing assumes that the specific logic is important and must be tested to guarantee to system proper functioning. The main use of the white box is in error based testing. In white box testing, you are looking for bugs that have a low probability of executing, have been carelessly implements, or were overlooked previously. One form of white box testing, called path testing, makes certain that each path in an object's method is executed at least once during testing.

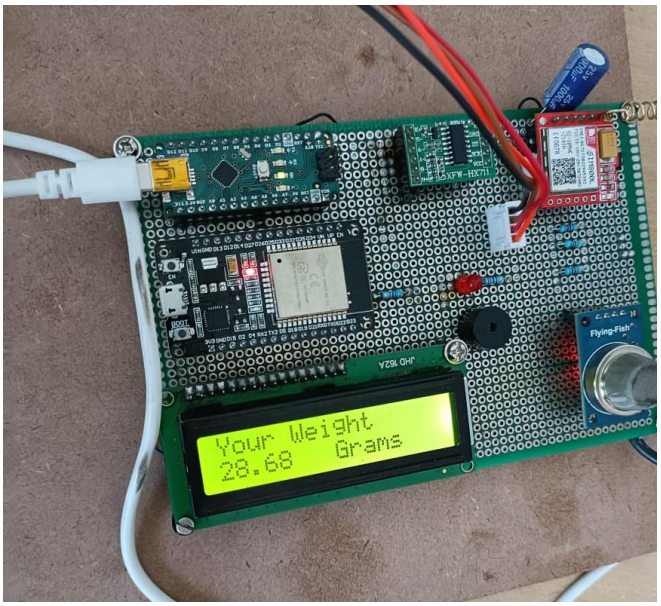
Security Testing

Attempts to verify the protection mechanisms built into the system.

Performance Testing

This method is designed to test runtime performance of software within the context ofan integrated system

9.RESULTS



If the LPG gas is going to leak, then the MQ-6 gas sensor will sense the LPG and butane gas concentrates in the air either at home or in industry, it will give a HIGH pulse on its DO pin and Arduino constantly reads its DO pin. When Arduino receives a HIGH pulse from the LPG Gas sensor module it displays that LPG Gas Leakage Alert message on a 16x2 LCD and stimulates a buzzer which beeps again until the gas detector module doesn't recognize the gas in the environment.

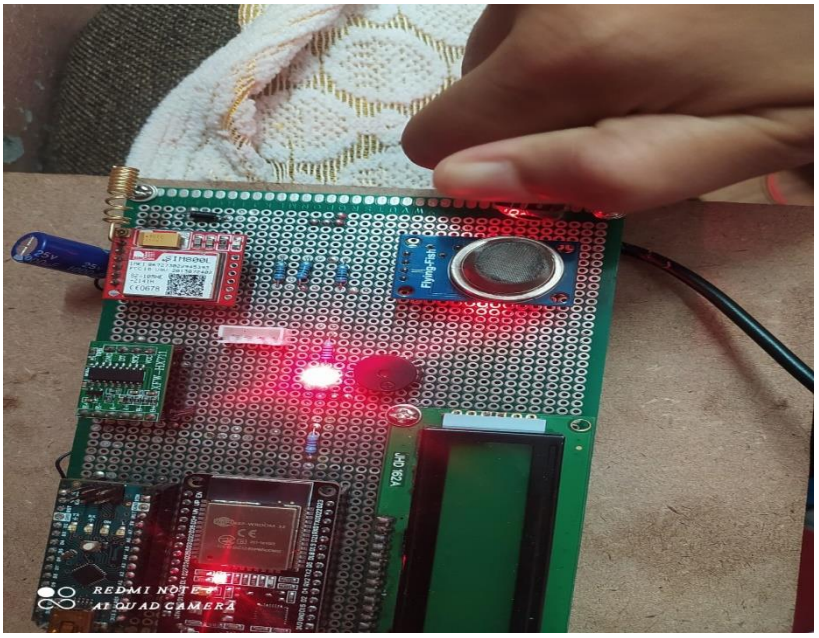
Arduino device will produce the buzzer sound based on the time duration by given instructions in code. When Arduino gets a LOW pulse from the LPG Gas detector module, then LCD will show that "No LPG Gas Leakage" alert message.

Arduino manages the complete process of this system like reading the LPG Gas sensor module output, sending a message to LCD, and stimulating the buzzer. We can set the sensitivity of this sensor module by an inbuilt potentiometer located on it.Load cell which is also known as a pressure sensor is used to detect the weight of the gas and the result will be displayed through an LCD display. GSM modem is capable of sending and receiving messages and the result will be displayed through LCD display.

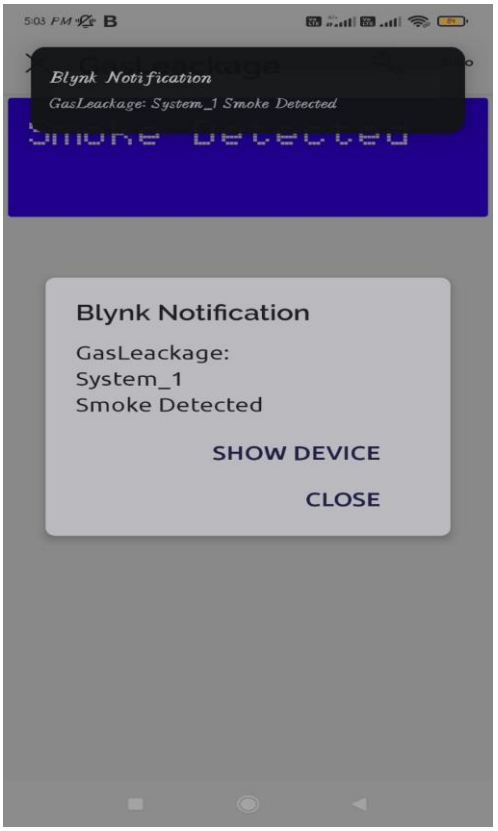
If the weight of the cylinder is below the threshold level, the booking confirmation message will be sent to the user through mobile. Message will be sent from the GSM module to the LPG agency and gets a return notification of when the LPG is delivered.

10. PROJECT OUTPUT:

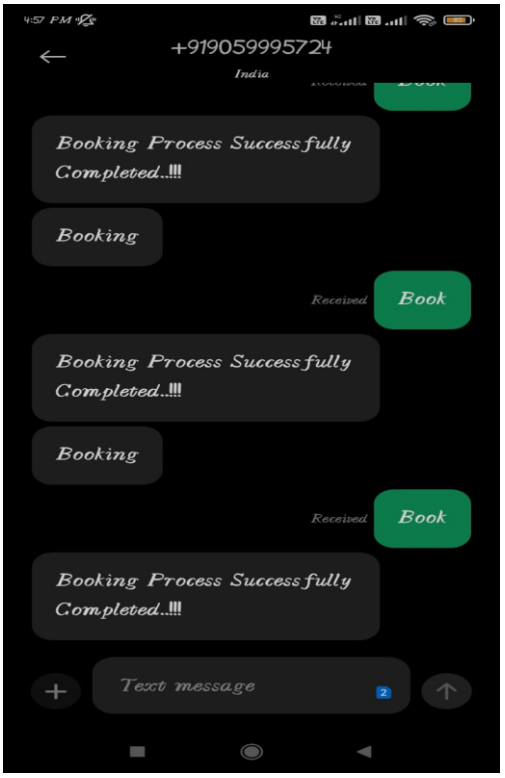
Buzzer will produces beep sounds when LPG gas is detected by MQ-6 sensor and this resultant is indicated by using LED blinking.



The notification alert is also sent to the registered mobile number through GSM module when the LPG gas leakage is detected by MQ-6 senssor.



The notification is sent when the LPG gas is going to complete or LPG weight reaches to thresh hold value and asks user permission to automatic book the new LPG cylinder through the GSM module to registered agency contact number.



10. CONCLUSION

The major motive here is to bring forth safety to the users of LPG (Liquefied Petroleum Gas) in various sectors like cooking, automobiles, industries, etc. A budget-friendly gas leakage detection system was put forward, mapped out, and successfully executed in this paper. It uses an MQ-6 sensor and load cell to monitor the LPG being used completely to prevent accidents caused by carelessness or misuse of LPG. In addition to gas leakage detection, this system gives an entirely mechanized approach to gas booking. The cost incriminated in developing the system is outstandingly low and is much less than the cost of gas detectors economically obtainable in the market.

11. REFERENCES

[1] Mahalingam, A., R. T. Naayagi, and N. E. Mastorakis. "Design and implementation of an economic gas leakage detector." Recent Researches in Applications of Electrical and Computer Engineering, pp. 20-24, 2012.

[2] Attia, Hussain A., and Halah Y. Ali. "Electronic Design of Liquefied Petroleum Gas Leakage Monitoring, Alarm, and Protection System Based on Discrete Components." International Journal of Applied Engineering Research, vol. 11, no. 19, pp. 9721-9726, 2016.

[3] Apeh, S. T., K. B. Erameh, and U. Iruansi. "Design and Development of Kitchen Gas Leakage Detection and Automatic Gas Shut off System." Journal of Emerging Trends in Engineering and Applied Sciences, vol. 5, no. 3, pp. 222-228, 2014.

[4] T.Soundarya, J.V. Anchitaalagammai, G. Deepa Priya, S.S. Karthick Kumar, "C-Leakage: Cylinder LPG Gas Leakage Detection for Home Safety," IOSR Journal of Electronics and Communication Engineering, vol. 9, no. 1, Ver. VI, pp. 53-58, Feb. 2014.

[5] Ashish Shrivastava, Ratnesh Prabhaker, Rajeev Kumar, Rahul Verma, "GSM based gas leakage detection system." International Journal of Emerging Trends in Electrical and Electronics, vol. 3, no. 2, pp. 42-45, 2013.

[6] Kaushik Vipul R. et al., "IOT-based energy meter billing and monitoring system -A case study," International Research Journal of Advanced Engineering and Science, Volume 2, Issue 4, pp. 64-68, 2017.

[7] V. N. Patil et al., "Criminal Identification Using Arm7," International Research Journal of Engineering and Technology, Vol: 04, Issue: 3, pp.677- 680, Mar -2017.