BINDURA UNIVERSITY OF SCIENCE EDUCATION

DEPARTMENT OF ELECTRONIC ENGINEERING

FINAL YEAR PROJECT

2024-2025

TITLE:

REMOVAL OF REDUNDANCE IN FUEL PROCUREMENT IN PRIVATE ORGANISATION

NAME: **KELVIN MAKWARIMBA**

REG. NUMBER: **B202666B**

SUPERVISOR: **Eng KOMICHI**

DECLARATION FORM

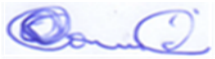
"I, Kelvin Makwarimba, hereby declare that this project report entitled 'Removal Of Redundance in Fuel Procurement in Private Organisation' is the result of my own original work, carried out under the supervision of Engineer Sylvanus Komichi, in partial fulfilment of the requirements for the Bachelor of Science Honors Degree in Electronic Engineering at the Bindura University of Science Education.

This work has not been submitted for any other degree or examination at any other institution of higher learning, and all sources used have been acknowledged accordingly."

10/07/2025

....

(Signature of student) (Date)



10/07/2025

....

(Signature of Supervisor) (Date)

....

(Signature of the chairperson) (Date)

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to the Almighty God for granting me the strength, wisdom, and perseverance throughout the course of this project. Without His divine guidance and grace, this work would not have been possible.

My sincere appreciation to my supervisor, Engineer Sylvanus Komichi, for his consistent support, encouragement, and insightful guidance. His expertise and constructive feedback played a crucial role in shaping this project.

Special thanks go to my colleagues and classmates, whose assistance to some challenges, moral support and motivation were invaluable to the successful completion of this project.

I also wish to acknowledge the technicians at Bindura University of Science Education for their practical support, assistance in the laboratories, and help in sourcing components during the hardware development phase.

Lastly and most importantly, I am immensely grateful to my family for their unwavering love, patience, financial, and moral support throughout this academic journey. Their encouragement gave me the strength to overcome the challenges I faced during this project.

ABSTRACT

Fuel management in private firms in Zimbabwe has previously relied on multi-level authorization and paper-based manual systems with the use of fuel coupons, which has led to inefficiency, delays, and vulnerability to fuel theft and abuse. This project proposes and implements an automated, centrally controlled fuel purchasing system to address these problems. The system integrates user authentication, online monitoring of fuel levels, and controlled dispensing through embedded systems and a cloudless local database system. Ethereal to it is an ESP32 microcontroller tied with a keypad, ultrasonic level sensor, LCD display, and a pump which is solenoid valve operated, all functioning together to manage safe and priority access to fuel resources. The users are recognized by their unique ID and PIN, and his/her fuel allocation is verified from a MySQL database which is accessed using PHP scripts. Dispensing is authorized only if users meet allocation and priority requirements. The system keeps the database up to date to reflect outstanding allocations and keeps all transactions, to avoid overuse and enable accountability. The solution is advantageous in that it significantly improves efficiency, reduces operation downtime, reduces human error, and improves fuel distribution transparency, and thus constitutes a sustainable and scalable model of digital resource management for developing environments.

Contents

1 CHAPTER 1: INTRODUCTION

1.1 Background

Fuel procurement and management are at the core of the operational efficiency of any company employing vehicles, machines, or equipment that consumes fuel. In today's competitive high-velocity business world, having the ability to manage fuel resources can be a source of significant difference to the productivity, cost-effectiveness, and success of an organization. Organizations particularly from the developing countries like Zimbabwe, continue to use manual and archaic fuel management systems. They are typically characterized by inefficiency, errors, and lack of transparency that lead to wastage, stealing of fuel, and delays in work.

Fuel procurement processes were always paper based with manual workflows, coupon schemes and multiple levels of authorization, not only are such workflows time-consuming but also prone to errors, which makes it difficult to accurately monitor fuel consumption, monitor the levels of fuel storage tanks, and efficiently manage resources. Workers must navigate long bureaucratic processes to obtain fuel approval, resulting in unnecessary downtime and low productivity levels, for instance the writer during attachment at the Zimbabwe Broadcasting Corporation (ZBC), observed that fuel authorization had to go through tiers of offices, including the director, manager, and transport manager of the department, before it could be finally picked up by a coupon at the dispatch office. This prolonged process would lead to delays if there were no outlets where one could proceed, coupons did not exist, or fuel was not available. Such inefficiencies can have a chain reaction on the functioning of an organization, particularly for companies like the media, where immediate access to fuel is critical for news reporters to gather images and meet deadlines.

The inefficiencies in the manual systems of fuel management are also compounded by the lack of monitoring and prioritization in real-time. Without a centralized system to monitor the levels of fuel and usage, organizations lack the visibility they require to make effective decisions on fuel allocation. The consequence is that the most important equipment or vehicles end up going without fuel while the secondary operations burn disproportionate amounts of the resources. Moreover, the absence of automated low fuel level warnings can result in abrupt drainages, affecting operations and incurring additional costs.

These drawbacks must be minimized by use of an automatic prioritized mechanism of fuel acquisition and tracking. It would enhance the effectiveness of fuel management operations,

reduce the importance of manual intervention, and enhance transparency and accountability. Leveraging such technology can make organizations be able to move away from conventional paper-based systems and embrace a central computerized system with real-time data on fuel levels, usage patterns, and priority allocations. This would essentially not only improve operational efficiency but also enable informed decision-making, prevent fuel wastage, and theft.

Fuel management has long been Zimbabwean businesses' biggest operational concern, particularly for those in vehicle- and machinery-intensive sectors. Historically, the country's private companies relied on contract schemes with contracted service stations to provide their fuel needs. Although such an approach was easy, it was not economical due to fluctuating fuel prices, third-party supplier margins, and transport inefficiencies. Companies typically faced lag time because drivers would take round-trip excursions to and from third-party service facilities, falling behind in production time. Companies looked for measures to reduce their costs and gain more control of their fuel supply chains over time. One of the options was that private company-owned service stations be established. Although this arrangement eliminated dependency on external suppliers and negated traveling inefficiencies, it introduced a complete set of issues, such as the need for storage structures, maintenance costs, and fuel distribution making intra-management cumbersome.

Efforts to mitigate such issues lead to majority of organizations to resort to a hybrid setup whereby they bought fuel in bulk from national majors like the National Oil Company of Zimbabwe (NOCZIM) and supplied it internally using a paper coupon system. Systems such as these where departments or individuals requiring fuel would be given coupons through a multi-step authorization process, which they would then exchange in place of fuel at the company store facility. While this system was aimed at keeping costs under check and enhancing controls, it came to prove serious deficiencies within no time. The coupon system, while ingenious the concept of fuel dispensation with silky flow, proved to be a snag because it was based on mechanical interferences. Workers were usually compelled to go through a maze of approvals—circling the offices of various directors, managers, and transport officers for signatures—before they could finally proceed to retrieve coupons from a dispatch office. The cumbersome system, observed by the author on industrial attachment at the Zimbabwe Broadcasting Corporation (ZBC), often led to paralysis of operations. News reporters tasked with covering time-sensitive events would be delayed in receiving fuel, jeopardizing their ability to take crucial footage. Similarly, employees who depended on company transport for

daily use were stranded when fuel coupons could not be located or approving personnel were unavailable.

The inefficiencies of the coupon system were further exacerbated by Zimbabwe's broader economic challenges, including frequent fuel shortages and currency volatility. These external pressures made it difficult to plan fuel since organizations struggled to balance constrained reserves against fluctuating demand. Without real-time visibility into fuel balances or rates of consumption, businesses over-ordered fuel when it was in tight supply (high cost) or under-ordered with resultant abrupt shortages that caused disruptions in operations. Dispatch teams would occasionally discover that the storage tanks had been depleted after coupons had been handed out, making crucial machinery idle and broadcasts late. Secondly, the manuality of the system left room for malpractice, i.e., theft of coupons, forgery, or misallocation, which added to organizational loads. Additionally the lack of prioritization in fuel allocation. At most firms, fuelling was on a "first-come, first-served" system, irrespective of relative urgency of vehicles or equipment to critical operations. As an illustration, a generator at a media firm could be given lower priority than a fleet of employee transportation vehicles, though the failure of the generator would bring down entire production lines. Such fuel supply and operational need disconnection frequently meant expensive downtime for capital assets.

Current system pain points point to an urgent necessity for fuel management innovation. While certain Zimbabwean firms have begun to adopt digital solutions for logistics and inventories, fuel procurement continues to be based on old ways. This gap represents a challenge to leverage technology in order to exploit inefficiencies at the system level, reduce reliance on labour-intensive, fault-prone processes, and make fuel management an extension of organizational goals. The ZBC and other such experiences present a compelling case for redesigning fuel procurement—not as a logistical procedure, but as a strategic one that has a direct effect upon productivity, cost control, and business resistance. By using automation, real-time tracking and priority distribution

1.2 Problem statement

Inefficiency of the system due to the absence of automation and first come first serve system of the coupon system without priority makes the system inefficient. Lack of knowledge regarding our own resources also renders it impossible to plan leading to under or over purchasing fuel and being vulnerable to theft.

1.3 Aim

To design a fuel filling system which maintains a record of our fuel

1.4 Objectives

 To create a database of all the fuel users within the organization and allocate priority level to each user based on importance and quantity allocated.

 To create user authentication and verification system via database

 To create a fuel dispensing system

 To create a user-friendly interface

1.5 Problem solution

The proposed solution is to create an Automated filing system which will have all the users in a centralized database that will contain their ID and pins and fuel allocation for each of the users this system also allows the administrator to view and edit the number of letters that the user can be given. The database will also update the number of letters after the user has filled their tank. The system will encompasses use the validation and validation using a keypad via via a localhost. Dispensing fuel the user wants if the number of letters that they enter is less than or equal to the number of liters that they have been allocated in the database if not it won't and after dispensing it will update in the database the remaining number of liters for the given user

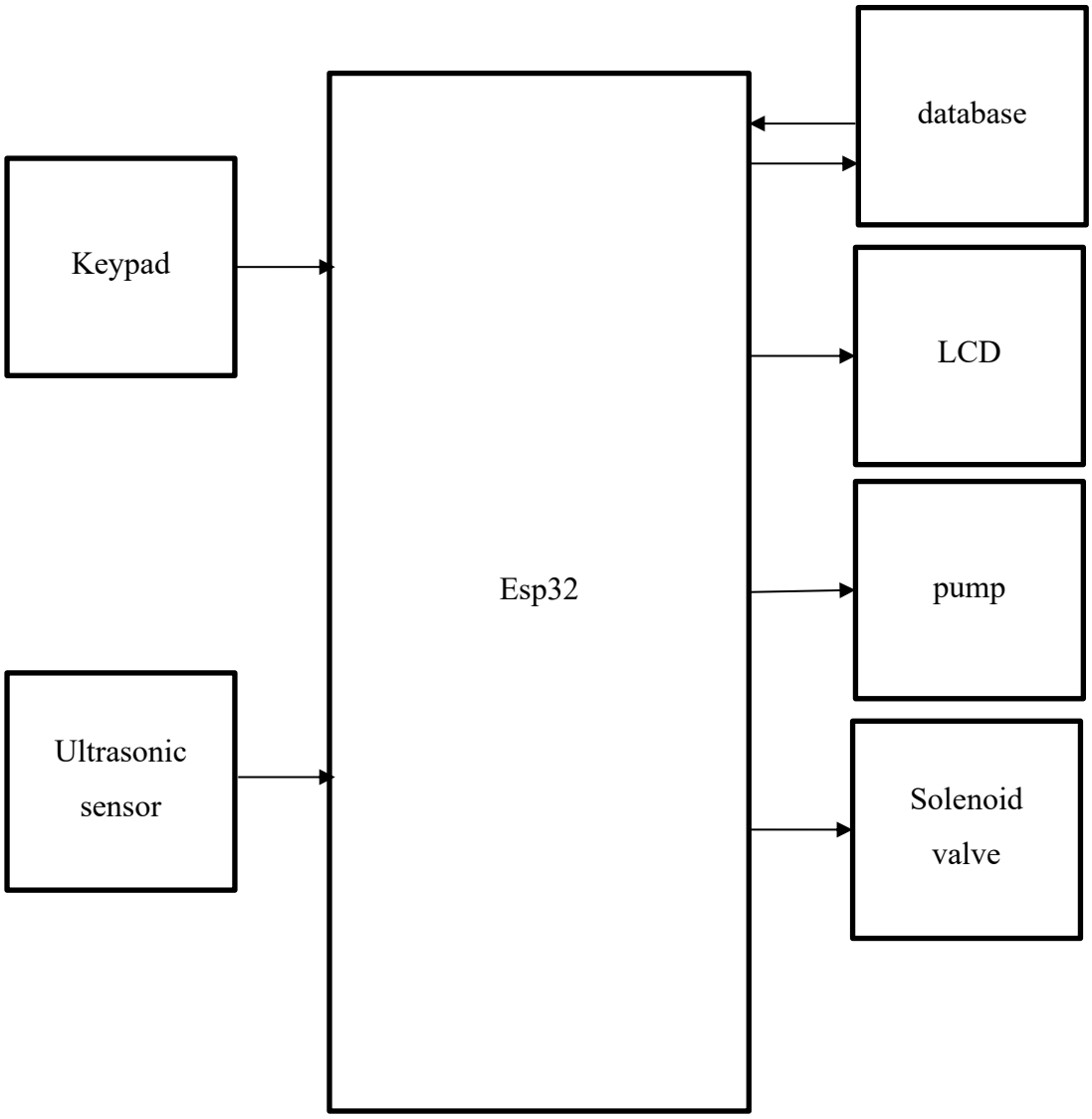


Figure 1.1: System block diagram

Figure 1.1 shows the logical interconnection of all the components of the system are used for the authentication of the user by entering the id and password from the keypad. The level sensor will be used to check the level fuel will be stored in the reserve tank; the database will store all the information like machinery id and priority level and LCD and LED will be used to show the status of the tank. The pump and solenoid valves will be used to pump the fill and open and close the valve respectively.

All equipment will have a unique id and pin that will be entered by the user via a keypad to verify that you are the owner. The id and pin will be compared with that in the database before verifying the priority level of your specified equipment to check if it can fill. The system will

display the litters allocated to that individual. After it has displayed the litters allocated to that individual, it will prompt you to enter the quantity of litters which you require, and if the quantity of litters is either below or equal to the quantity in the database and the level of priority is in accordance with the status of the tank, then the system will allow you to fill your device. After dispensing the fuel the database will be updated on the filling.

1.6 Conclusion

Manual fuel management systems in Zimbabwe, like those in institutions such as ZBC, highlight operational inefficiencies by bureaucratic delays, human error, and lax monitoring. These inefficiencies fuel waste, theft, and downtime, especially in time-sensitive sectors such as media.

These gaps are addressed by the proposed electronic priority-based system through automation, a centralized priority database, and IoT-enabled alerts. Prioritization of critical assets like broadcast generators during shortages, coupled with user authentication, enhances accountability and reduces misuse. Real-time fuel data enables informed decision-making, procurement optimization, and minimization of shortages.

Applying this system offers a strong, cost-effective solution for private Zimbabwean businesses amidst fuel scarcity. Through the replacement of outdated methods with an open, priority-driven system, downtime is minimized and strategic resource alignment is achieved. This system promotes sustainable fuel usage and operational efficiency, positioning businesses for growth.

2 Chapter 2: LITERATURE REVIEW

2.1 Introduction

Coupon systems have been widely used in numerous sectors, especially in rationing limited resources such as fuel. The majority of these systems involve distributing paper-based or physical coupons to users, which can be exchanged for fuel. They are low-cost with straightforward processes to monitor distribution. Among their drawbacks, however, is vulnerability to fraud, theft, and inefficiency in record keeping, especially in the case of large user numbers. Coupon systems also lack the flexibility to accommodate real-time updates.

Literature in couponing systems points out the need for more efficient, transparent, and scalable methods of resource allocation. Authors have explored the inefficiency of physical coupon distribution and the possibility of a move towards electronic systems, which can offer higher flexibility and reduce errors on the part of humans [1]. The use of an electronic system founded upon a centralized database circumvents most limitations of traditional coupon systems. There has been research conducted in the development and design of such systems for fuel procurement industries, food distribution industries, and other resource-based management. The user information can be stored and managed by centralized databases, making it easy to track the fuel requirement and allocation for users or departments within an organization. Research describes the advantage of centralized databases in managing large amounts of data and minimizing administrative burdens [2]. Electronic database can improve record keeping, prevent errors, and facilitate easy updates in real time.

The distribution of fuel can be optimized by developing a mechanism that prioritizes different users based on their requirements for the fuel. Different optimization models, including algorithms for dynamic prioritization, have been proposed in the literature. Algorithms like Round-Robin Scheduling and Priority Queue Scheduling are used heavily in systems where resources need to be allocated to users based on priority. These algorithms ensure that people with urgent or critical needs (e.g., emergency services) are given priority, while others can be queued [3].

The use of priority scheduling in fuel procurement can maximize utility while being equitable and optimize the allocation of resources. Adaptive models are also proposed in some research, which alter the priority levels dynamically based on the usage pattern and fuel availability [4].

Such dynamic adjustments can ensure users' demands are met at all times in an effective and equitable way.

The cornerstone of the proposed electronic system is the database. Modelling databases that can handle user details, fuel assignments, and transactions for the system to work efficiently is essential. Database system literature will focus on aspects such as data integrity, security, scalability, and performance tuning. Relational databases such as MySQL or PostgreSQL have been widely used to manage transactional systems. These systems use structured query language (SQL) to query and manage data, which is ideal for structured data like user profiles, fuel requirements, and transaction history [5]. Relational models can also provide excellent support for data integrity, with fuel data being consistent and correct across the system.

Recently in the last few years, cloud computing has become an increasingly viable choice for dealing with huge volumes of data. Cloud-based technologies, such as Amazon RDS or Google Cloud SQL, have the potential to provide scalability, flexibility, and high availability, making them very attractive for fuel procurement systems with a large number of users [6]. Cloud-based solutions also support integration with mobile applications, which enables customers to see their fuel allocation in real time from any place.

While relational databases can be a default for structured data, NoSQL (Not Only SQL) databases such as MongoDB or Cassandra can be useful in cases where data structure can change in the future. A NoSQL (Not Only SQL) solution would be ideal to manage dynamic and unstructured data, especially in the early stages of the system where fuel distribution and user requirements can evolve rapidly [7]. Since system will be dealing with sensitive user data and fuel transactions, security is of prime concern. Various studies in the field of electronic procurement systems have spoken about how user data can be secured and fraud prevented.

Role-based access control (RBAC) and multi-factor authentication (MFA) are generally implemented to ensure that only authorized personnel are allowed to view or modify critical system data [8]. These technologies can secure unauthorized access to fuel and protect against insider threat. Machine learning models, in particular anomaly detection algorithms, function in other industries to identify unusual patterns of behaviour that can indicate fraudulent activity [9]. These models can monitor transaction logs for anomalies, making the system secure and trustworthy. The other most important aspect of the proposed system is the user interface (UI) and user experience (UX) design. The literature emphasizes the importance of making the system easy to use, even for those with limited technical know-how [10]. Interactive

dashboards that allow users to visualize their fuel consumption and order fuel efficiently can be used to enhance the user experience. Furthermore the responsive mobile apps and web interfaces must be designed with simplicity in mind for accessibility and usability by workers of different functions within the company. The chosen design will utilize MySQL for the creation of the database. phpMyAdmin will handle all the activities that will be associated with the database operations

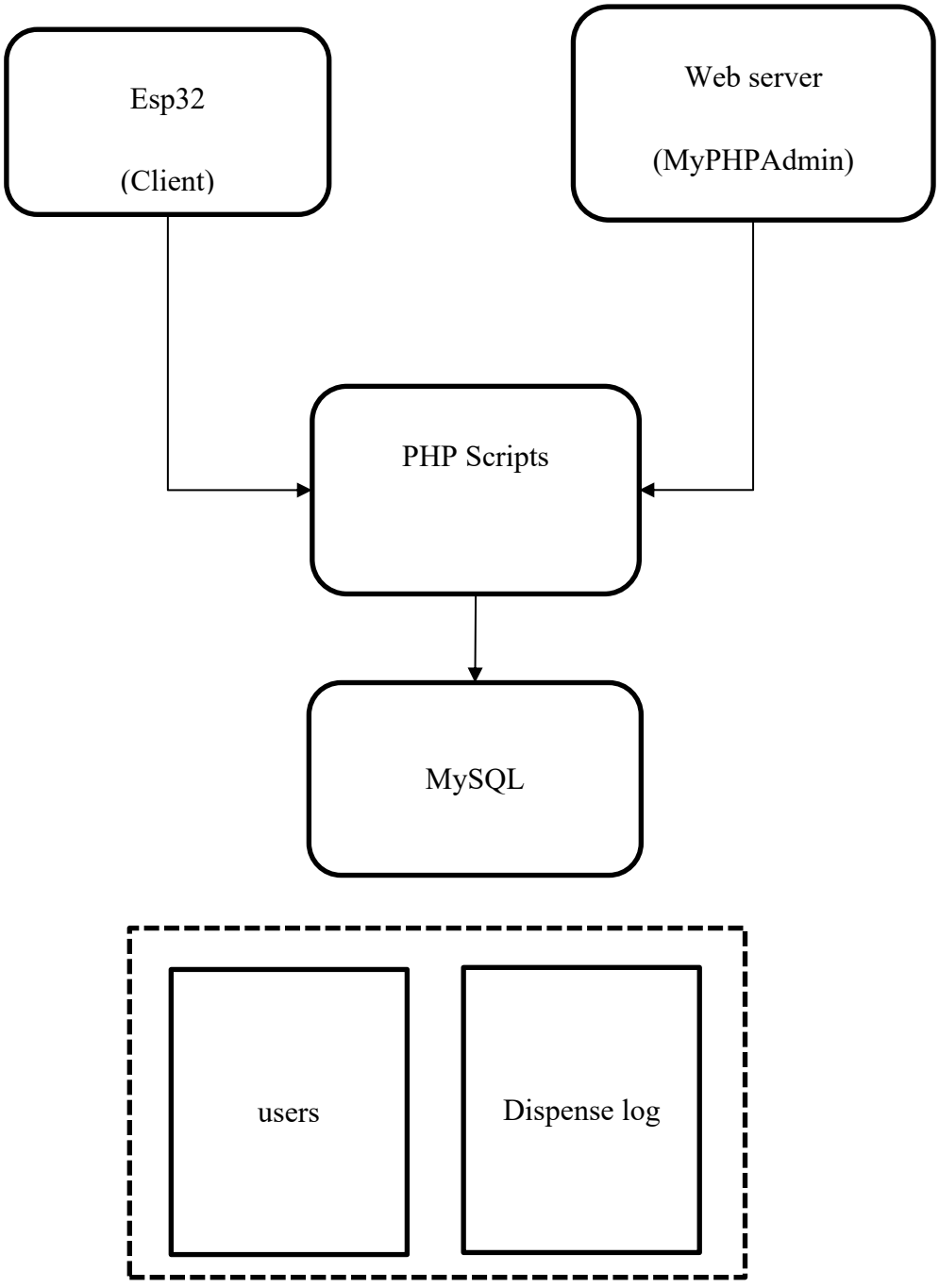


Figure 2.1:functional block diagram of the database management system

Figure 2.1 shows the structure of our database and how to be set up with phpMyAdmin server that will be used to manage the database. The ESP 32 that will be acting as the client will make requests to forward phpMyAdmin server then the phpMyAdmin server will you execute the command on the local host and then connect it to the database. There will be 2 tables in our database which are the users table and the dispense log table the user table will be utilized to hold user data and the dispense table be utilized to show every and each dispensing that has occurred.

Several organizations have taken up comparable systems for fuel purchasing or resource assignment in other industries. Companies in the transportation and logistics sectors have switched to computerized fuel distribution and fuel consumption management systems. Case studies of these industries could provide real-world lessons in best practices for database design, user management, and fuel distribution. The Oil & Gas Industry regularly uses complete software solutions to monitor fuel consumption by multiple vehicles and perform procurement in a controlled manner [11]. They employ real-time data and advanced forecasting models to predict fuel needs and optimize distribution to provide fair and priority-based allocation of resources.

2.2 Ultrasonic level sensor

Figure 2.2:Hr-sc04 ultrasonic sensors [12]

Figure 2.2 shows an ultrasonic level sensor is based on the time-of-flight (TOF) principle and measures distance to a target using high-frequency sound waves (typically 40 kHz). The sensor emits the short ultrasonic pulse from the transmitter that travels through the air, reflects off the target surface (solid or liquid), and returns to the sensor. The same transducer receives the echo

pulse and converts it back into an electrical signal. The sensor calculates the distance based on the time delay between the pulse transmitted and the echo received. The relationship is given by the formula:

$$distance = \frac{speed\ of\ sound \times time\ delay}{2}$$

This formula is taking into account the two-way traveling of the sound wave. The speed of sound is approximately 343 m/s at 20°C but varies with temperature. High-end ultrasonic sensors have temperature compensation mechanisms built in for precision[13]. These sensors are non-contact and therefore ideal to measure in hostile environments, i.e., corrosive or viscous liquids, and prevent fouling issues inherent in mechanical sensors [14].

Pin Description (HC-SR04 Ultrasonic Sensor)

The HC-SR04 is a very standard ultrasonic sensor consisting of four primary pins:

VCC (Pin 1): Power input of 5V DC, which powers the sensor.

Trig (Pin 2): The trigger pin, through which pulses are initiated when a HIGH signal of 10 µs is applied.

Echo (Pin 3): Echo pin, which goes HIGH for the duration of the round-trip travel time of the ultrasonic pulse.

GND (Pin 4): GND pin, connected to the circuit ground.

2.3 RG1602A lcd with I2C Connector

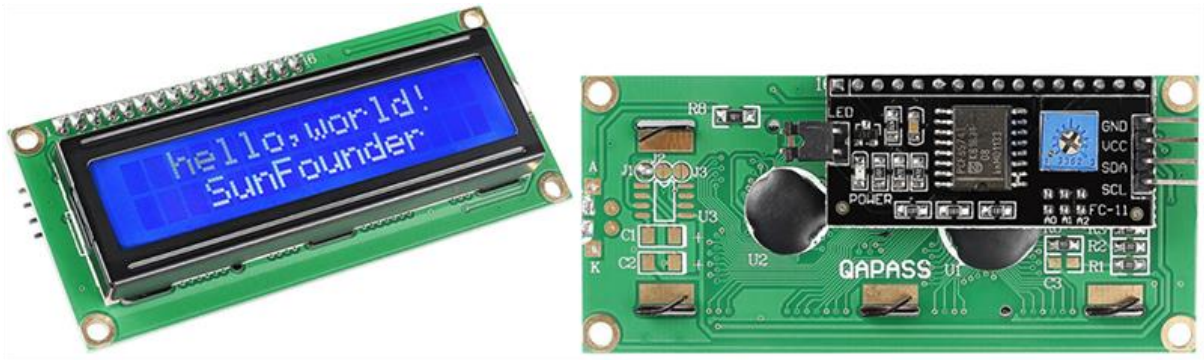


Figure 2.3:RG1602A LCD[15]

Figure 2.3 shows an RG1602A LCD is a 16x2 character display and has I2C (Inter-Integrated Circuit) communication, making it convenient to interface with microcontrollers like Arduino without using many pins. Benefits from both simplicity and low power usage. The I2C communication is convenient since it consumes fewer connections as it uses only two data lines SDA and SCL (data line and clock line) to drive the display.

Pin Description (RG1602A LCD with I2C) in figure 2.3

VCC (Pin 1): 5V power supply.

GND (Pin 2): Ground connection.

SDA (Pin 3): Serial Data pin for I2C communication.

SCL (Pin 4): Serial Clock pin for I2C communication.

A (Pin 5): Anode for the backlight (usually connected to 5V through a current-limiting resistor).

K (Pin 6): Cathode for the backlight (ground).

The I2C interface allows you to control the LCD with fewer pins and cuts down on wiring, which is especially helpful in projects with limited I/O pins.

2.4 Diode

Figure 2.4:diode

Figure 2.4 show a diode is a two-terminal semiconductor device that allows current to flow in one direction and blocks it in the opposite direction [16]. Diodes find basic applications in

electronics for rectification, voltage regulation, demodulation of signals, and circuit protection. The basic PN junction diode will conduct when the anode potential is greater than the cathode, and special devices like Zener diodes utilize breakdown characteristics for voltage regulation [17]. Embedded systems in diodes protect sensitive circuits from voltage spikes or reverse polarity, contributing significantly to system reliability.

2.5 Pump

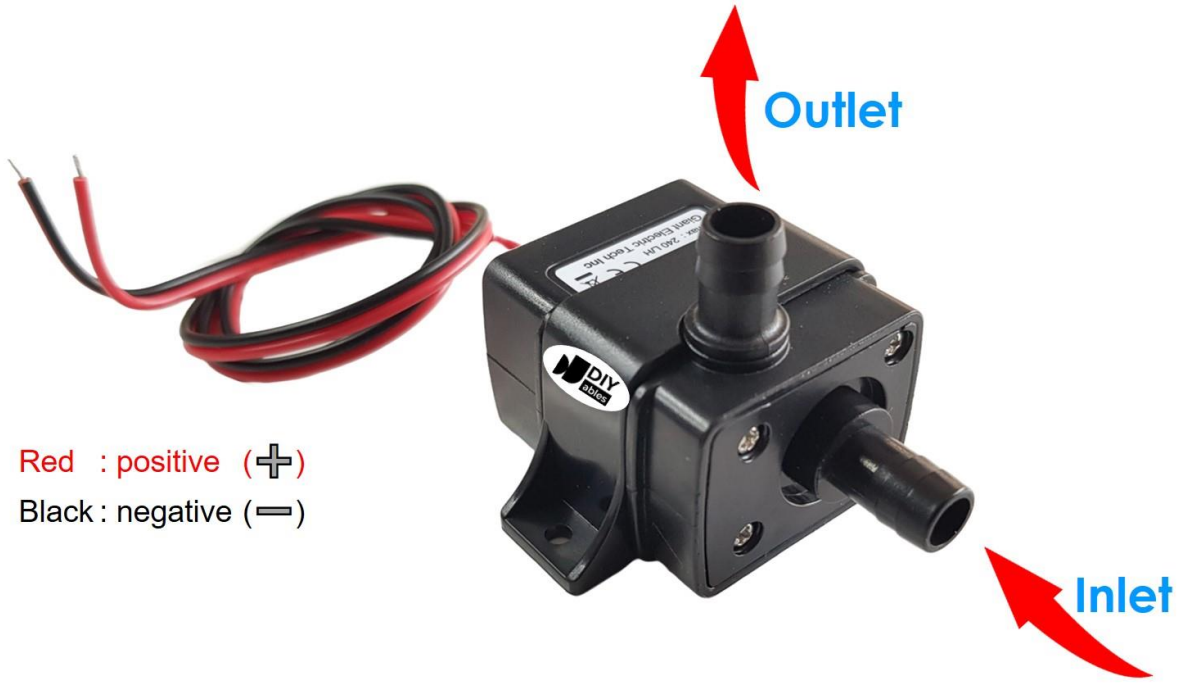


Figure 2.5:12V submissive pump

Figure 2.5shows a pump which is a mechanical device used to move fluids by increasing their pressure or flow rate [18]. Miniature pumps such as solenoid-operated diaphragm pumps or peristaltic pumps are utilized for precision liquid dispensing applications in automated systems[19]. These pumps are critical in applications like water purification, chemical dosing, and dispensing of fluids in a controlled manner. Electronic control of pumps makes it simple for them to be interfaced with microcontrollers to provide precise volume control when triggered by sensor input or user input. Pumps in automated liquid dispensing applications ensure that pre-set volumes of fluid are accurately dispensed

2.6 BC547 transistor

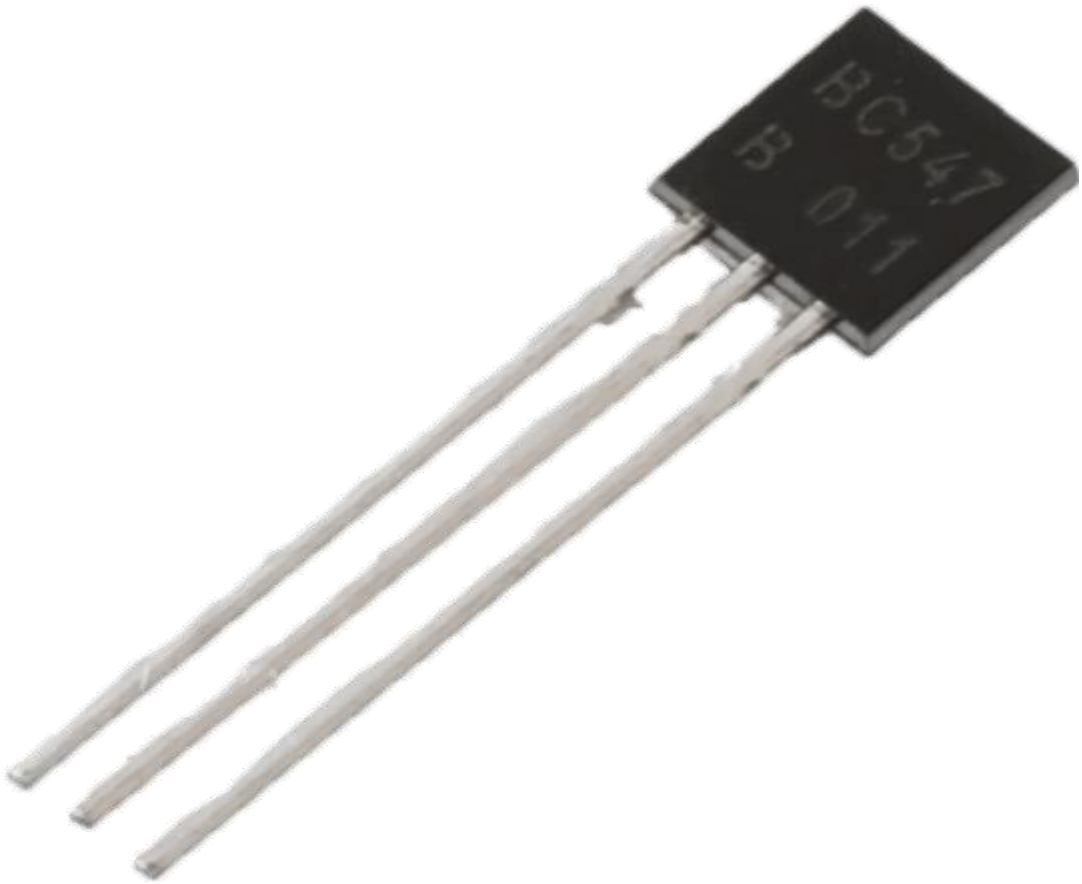


Figure 2.6: BC547 transistor

Figure 2.6 shows a BC547 which is three-terminal semiconductor devices that may be utilized as amplifiers or switches [20]. Since their invention in 1947, transistors have revolutionized electronics, substituted bulky vacuum tubes and enabling today's electronic products [21]. They are mainly categorized as Bipolar Junction Transistors (BJTs) and Field Effect Transistors (FETs) according to their characteristics. BJTs are current devices, while FETs are voltage devices [22]. Transistors in embedded systems have key functions in switching applications to allow microcontrollers to control higher-current devices such as pumps or relays. This switching capability provides electrical isolation and enables accurate control of power to external loads.

2.7 Resistor

Figure 2.7:Resistor

Figure 2.7 shows a resistor which is a passive electronic component that limit or control the flow of electric current in circuits by dissipating electrical energy in the form of heat [23]. They are defined in terms of their resistance value, power rating, and tolerance. Resistors have a broad area of applications that include current setting, voltage division, transistor biasing, and protection of circuits from excessive current levels. There are several kinds of resistors, including carbon film, metal film, wire-wound, and surface-mount resistors, which are selected based on application requirements. Accurate resistor values are required for analog and digital circuits for stable and reliable system performance [24].

2.8 Relay

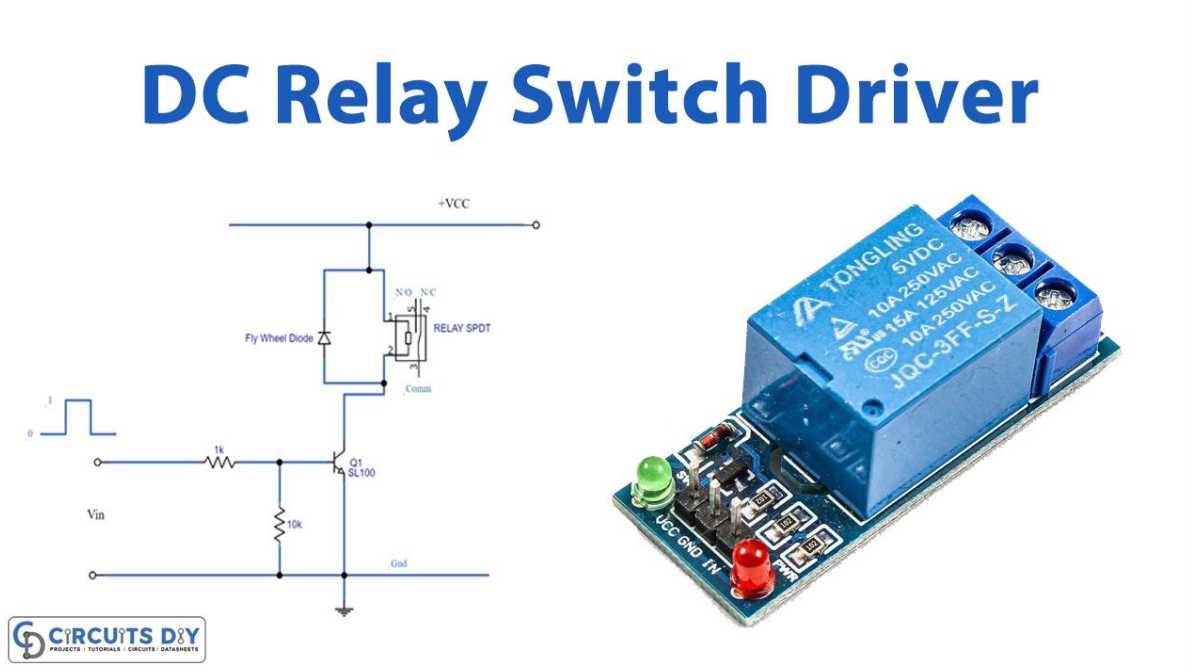


Figure 2.8:relay

Figure 2.8 shows a relay which is an electromagnetic switch that can employ a low-voltage signal to toggle higher-voltage or higher-current devices [25]. Driven by a 5V DC signal, the internal coil within the relay will create a magnetic field that will move internal contacts, thus switching another circuit on and off [26]. This makes the 5V relay especially well-suited for microcontroller projects, in which it can power devices such as pumps, lights, or motors safely while keeping high-voltage circuits isolated from sensitive electronics [27]. The majority of relay modules include protection devices such as flyback diodes to quench voltage spikes, ensuring smooth operation in embedded systems [28].

2.9 ESP32 Microcontroller

The ESP32, designed by Espressif Systems, is a highly versatile microcontroller that is highly utilized for Internet of Things (IoT) applications. It has a dual-core Xtensa LX6 processor with a clock rate of up to 240 MHz and is hence suitable for real-time applications such as robotics and industrial automation [29]. The ESP32 has built-in Wi-Fi and Bluetooth (BLE/Classic) support, reducing communication hardware requirements and simplifying IoT device development [30].

Figure 2.9:Esp32[31]

Figure 2.9 shows an esp32 (Espressif32) with peripherals such as 12-bit ADCs (Analog to Digital Converters), 8-bit DACs (Digital to Analog Converters, SPI, I2C, and PWM (Pulse Width Modulation), which enables the ESP32 to interface with a wide variety of sensors and actuators. This is particularly beneficial in environmental monitoring systems and predictive maintenance applications [32]. Besides, the ESP32's ability to manage power, with multiple modes of sleep, also makes it highly applicable to battery-powered devices, such as wearable [33]. To facilitate development, the ESP32 also supports several programming environments, such as the ESP-IDF (Espressif IoT Development Framework), Arduino IDE, and MicroPython, which further makes it accessible to professionals as well as hobbyists [34]. Furthermore, third-party platforms such as PlatformIO and libraries such as FreeRTOS are widely utilized to handle complex, real-time systems [35]. While the ESP32 is a versatile device with a wide range of applications, its ESP-IDF development platform is harder to learn compared to more introductory platforms like Arduino [36].

Applications in Electronic Engineering

1. IoT and Smart Systems: ESP32 is one of the best choices for IoT applications like intelligent agricultural systems, where it is paired with sensors like soil moisture sensors and LoRaWAN gateways for real-time monitoring [37].

2. Industrial Automation: Dual-core architecture in ESP32 supports predictive maintenance functions using vibration sensors to feed data into the microcontroller for analysis [38].

3. Robotics: Researchers use the ESP32 to drive robot motors and integrate it with SLAM algorithms in low-cost robotic systems [39].

4. Education: Due to its low cost and extensive documentation, the ESP32 is extremely popular in the educational segment for learning embedded systems and IoT concepts [40].

2.10 Conclusion

The transition from a coupon system of fuel procurement to an electronic system built on a centralized database is a key step in the modernization of the fuel resource management of a company. According to the literature, the creation of such a system will need to incorporate efficient user management, optimization of fuel allocation, robust database design, and high security level. Additionally, real-time prioritization mechanisms and fraud detection techniques will be integral in making the system secure and fair. With the chance to leverage

the advancement in database systems, cloud computing, and machine learning, companies can build a more efficient, scalable, and user-friendly electronic fuel procurement system.

3 CHAPTER 3: DESIGN

This chapter outlines the methods used in designing and developing the proposed priority-based fuel procurement and monitoring system for private institutions, it outlines the hardware and software used, the design architecture, data flow of all the subsystems. The chapter also outlines how the system addresses the problems highlighted in Chapter 1 and elaborates on the technologies examined within the literature review in Chapter 2.

3.1 System Overview

The system below integrates three primary components: user authentication and identification using keypad and database, fuel level detection using an ultrasonic sensor, and dispensing control using a solenoid valve and flow meter. The microcontroller (ESP32) is the device that controls the system, managing all the processes including communication with a remote database over Wi-Fi

3.2 System Description

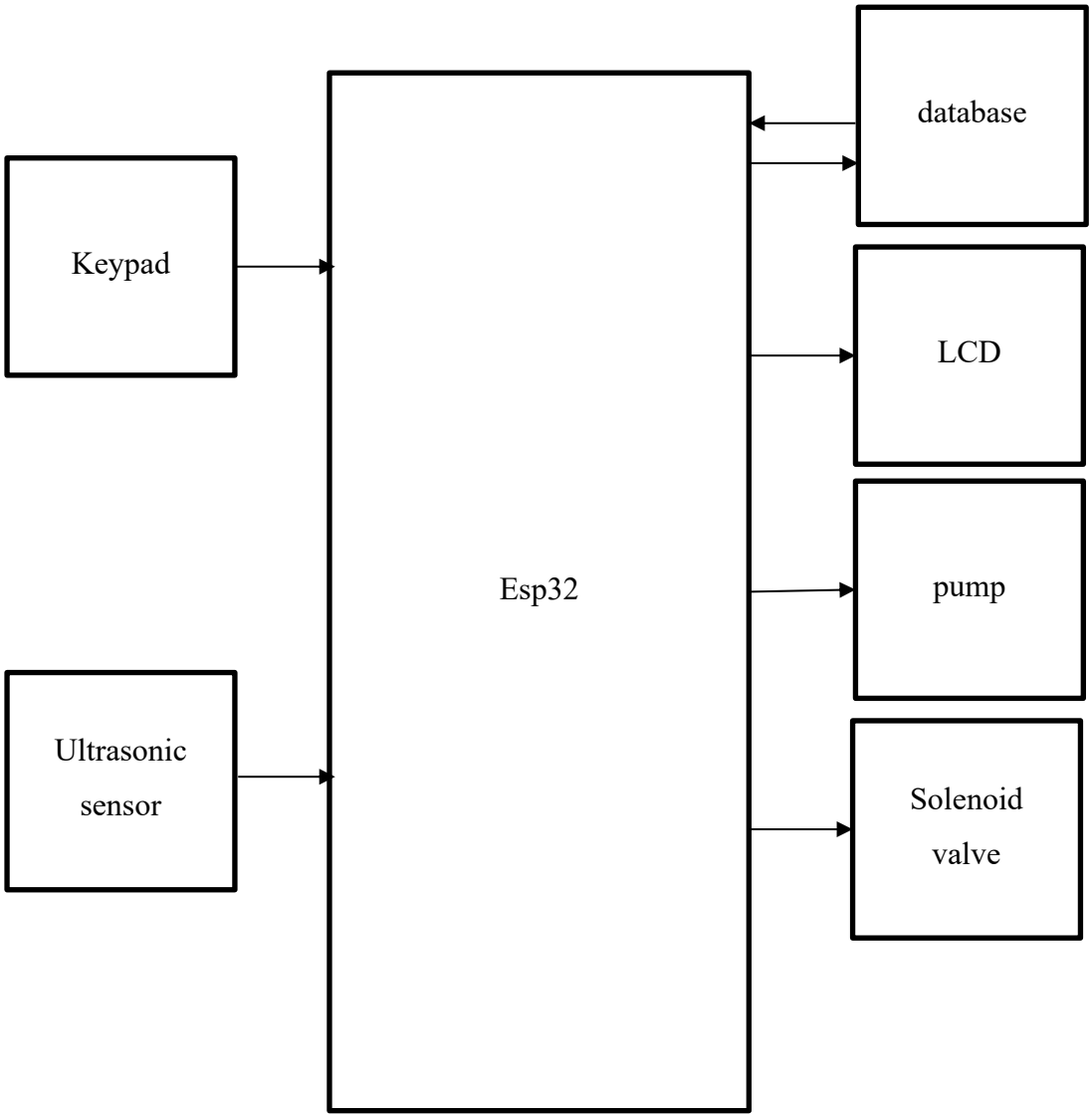


Figure 3.1: block diagram of the entire system

Figure 3.1 shows the block diagram of the entire system, showing the general connectivity of blocks to the ESP32. The diagram contains the following inputs 3x4 keypad for entering PIN, user ID, and required fuel quantity. Ultrasonic sensor for the measurement of the distance between the sensor and the fuel level, calculating the remaining tank volume. Database with user ID, PIN, allowable fuel quantity, and priority level. LCD display of the entire process, fuel level, user ID, and database updating. Pump for dispensing the fuel from the reserve tank to the user tank. Solenoid valve as a switch while dispensing fuel. LED indicator for three

different priority levels. This system integrates these components in order to control fuel dispensing based on user credentials and priority levels.

3.3 Design of different functionality in the system

Following is a list of functionalities that form the objective for this project the will focus on block diagrams, schematic circuits, flowcharts and breadboard testing

3.3.1 Wi-Fi connection and lcd functionality

Here in this part of the project, we are going to test our Wi-Fi connection and lcd display by a simple test of displaying whether our lcd can print connected if we are connected or printing connecting if it's not connecting. Below is the code for connecting to a Wi-Fi named Audit Training Workshop with no password. It will display "Connecting Wi-Fi" and at each half second, it's not connected it will and append a "." To the string for connecting Wi-Fi. There will not be a proteus for this code since we can't connect over Wi-Fi using a simulation

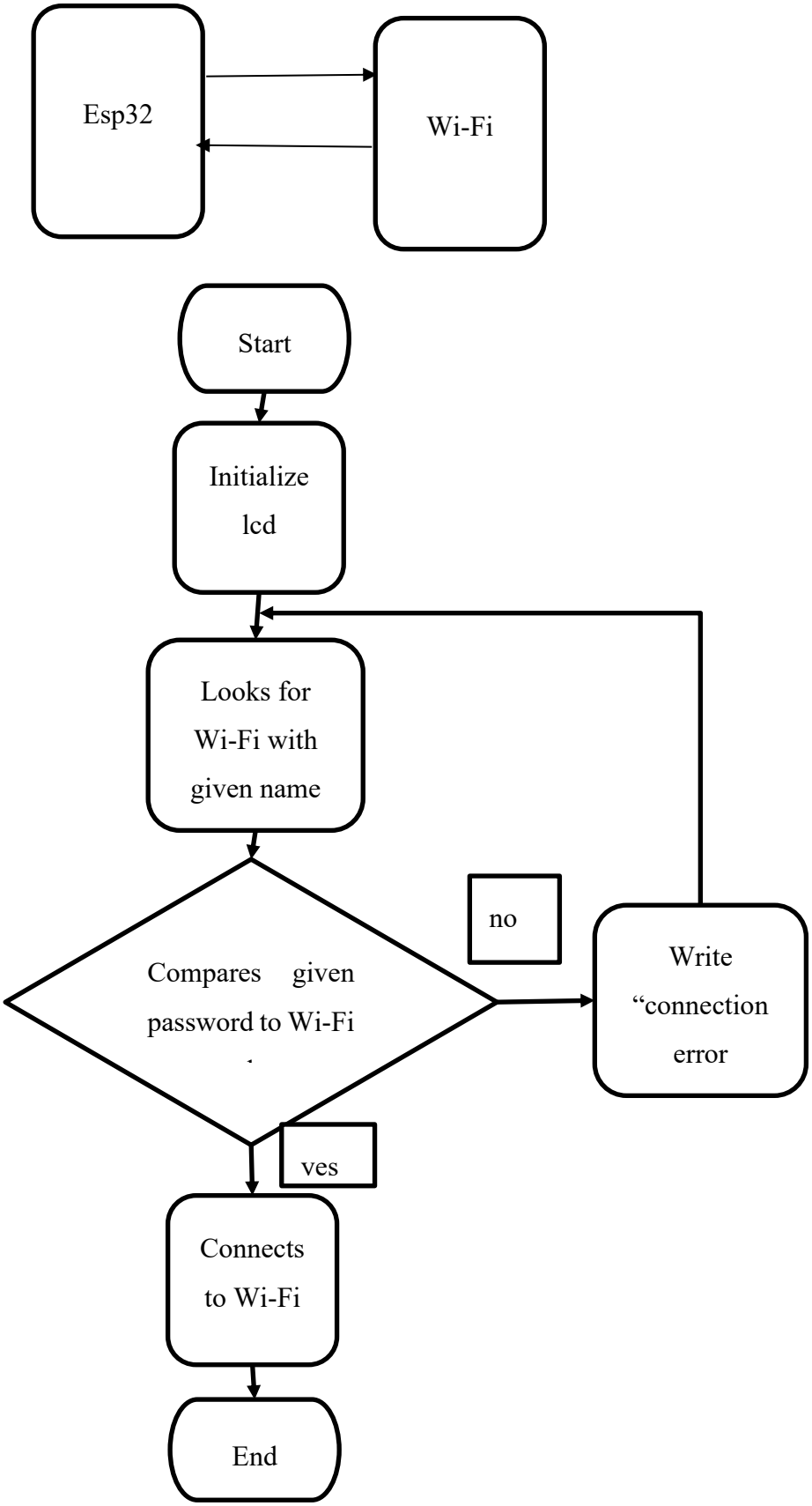


Figure 3.2: block diagram and flowchart for connecting to a Wi-Fi

Figure 3.2 shows the block diagram and flowchart for connecting to a Wi-Fi when the esp32 first initialize the lcd the checks whether the given password is in range then tries to connect to it by comparing the given password to the network password if not it returns a connection error if the Wi-Fi is not available. The code for connecting to the Wi-Fi is given below nnb

```
#include <WiFi.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
const char* ssid = "Audit Training Workshop";
const char* password = "";

void setup() {
  Serial.begin(115600);
  lcd.init();
  lcd.backlight();
  lcd.print("Connecting WiFi");
  Serial.print("Connecting WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    lcd.print(".");
      Serial.print(".");
  }
  lcd.clear();
delay(5000);
```

```
  lcd.print("WiFi Connected");
    Serial.print("WiFi connected");
}

void loop() {}
```

Figure 3.3: before connecting to Wi-Fi and after connected to Wi-Fi

Figure 3.3 shows the LCD display before and after connecting to a Wi-Fi from this our Wi-Fi our Wi-Fi and lcd test was a success

3.3.2 Connect database and get data.

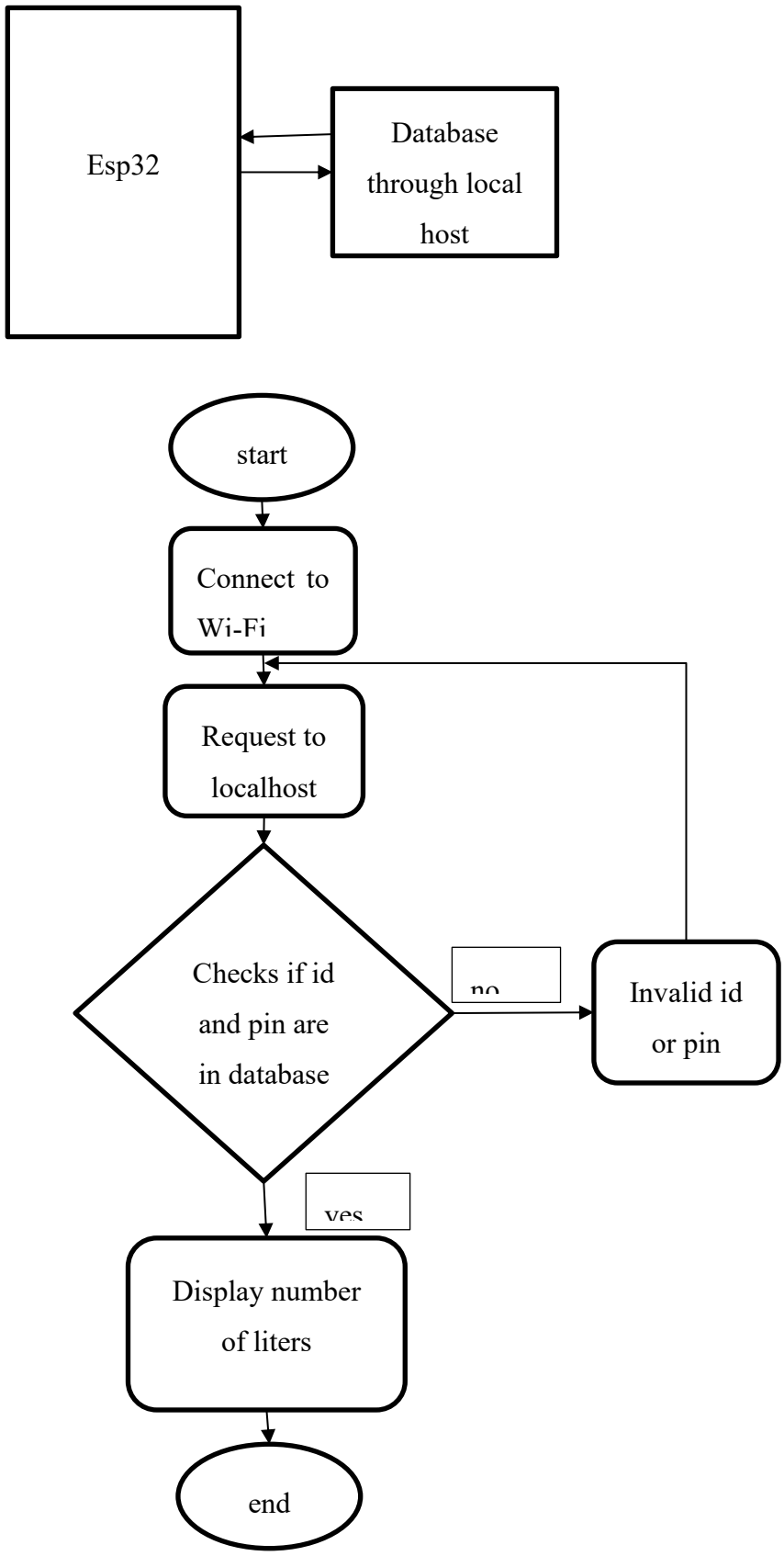


Figure 3.4: block and flow chart for retrieving allocated liters to an individual

Figure 3.4 shows the block in flow diagram for retrieving the number of liters allocated to a person in a database 18 encompasses the ESP 32 generating a URL that contains the location of the PHP file which will be run by the local host after running the command on the local host need to connect to the database and retrieve the number of liters stored in the database for the given you ID and pin then display it on the LCD monitor

Here in this example, the esp32 will connect to a database through a local host. This is achieved by the esp32 and the PC as the localhost must be on the same network (i.e. on the same Wi-Fi) to make communication achievable. There is a requirement for a local server to run in this instance we will use XAMMP server which requires Apache and MySQL modules to run because they are the one that will facilitate securing of a connection to database. The esp32 has an HTTP Client library that will be utilized to develop an API (Application Programming Interface) that will point to a php file that will run the code on the host machine. The code for database connection and printing remaining liters in the database is given below.

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "itel S24";
const char* password = "3ng1n33r1ng";
const String url = "http://192.168.251.117/verify_user.php?id=21477&pin=3296";//ip address of the server plus id and pin

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("WiFi connected");

  HTTPClient http;
  http.begin(url);
  int httpCode = http.GET();
  if (httpCode > 0) {
    String response = http.getString();
    Serial.println("Response: " + response);
  } else {
    Serial.println("HTTP GET failed");
  }
  http.end();
}

void loop() {}
```

The URL (http://192.168.43.122/verify_user.php?id=21477&pin=3296) has 3 parts to it which are the localhost IP address and this is 192.168.43.122 and it will seek for a file verify_user.php in the htdoc directory of the XAMPP installation directory, this makes it simulate as if the request was coming from inside. The last segment of the url is (id=21477&pin=3296) this segment carries 2 arguments which are pin and id these are to be authenticated in the database. Below is a php-code for connecting to database

User and password are for security and credential validation in the sense that we are using them as root and no password. The database we will be using is userinfo. $conn = new mysqli($host, $user, $pass, $dbname); - Creates a new MySQLi object and connects to the database. if ($conn->connect_error) {. } - Checks if there was any error in connecting the database. die ("Connection failed: ". $conn->connect_error); If it is a connection error, it displays an error

message and terminates the script. For fetching data input, we use $id = $_GET['id'] "; and $pin = $_GET['pin'] ???. To check input we use "$id = trim($id); and $pin = trim ($pin);. It then checks that neither the "id" nor "pin" value is empty with if (empty($id) || empty($pin)) {. }. It then sets a SQL query with placeholders for the "id" and "pin" values with $sql = "SELECT liters FROM userinfocred WHERE id = ? AND pin = ?";. where sensor _data is a table in the userinfo database. $stmt = $conn->prepare($sql); then prepares the SQL query for execution. 2. $stmt->execute(); executes the prepared query. For retrieving results we use $result = $stmt->get_result();. if ($row = $result->fetch_assoc()) {. }` Fetches the first row from the result set in this case it being liters. If there is no row returned liters being its name, it displays an error message. It then closes the prepared statement and database connection using $stmt->close(); and $conn->close();.

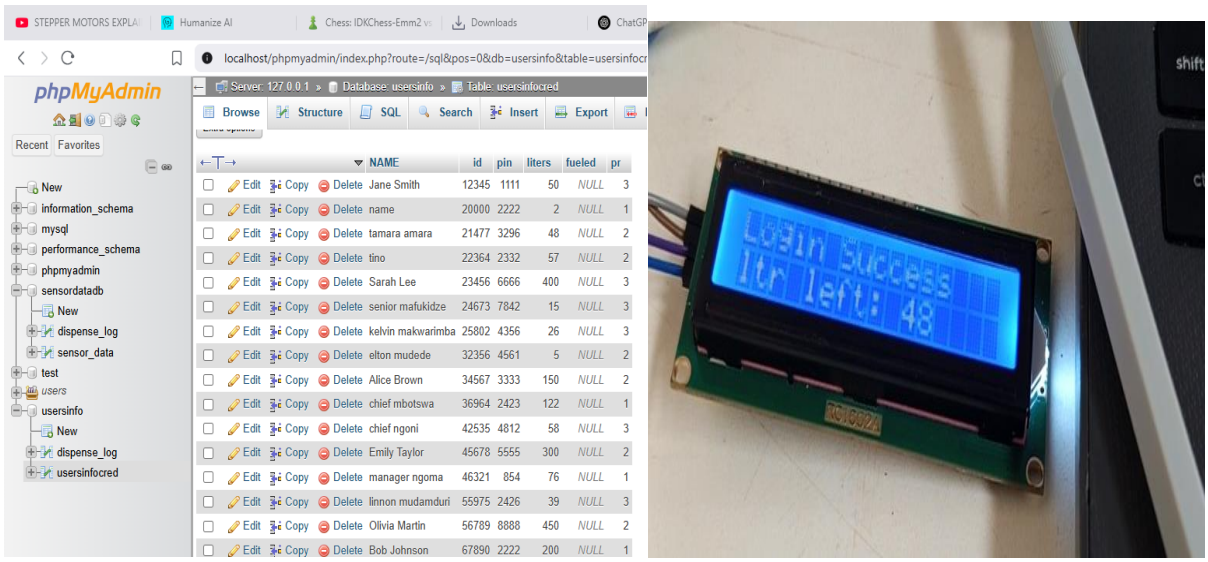


Figure 3.5:dislay of database(left) and results from URL test(right)

Figure 3.4 shows the database in PHP my admin on the left showing rows users in the database and on the right shows the display on the LCD of the number of liters allocated to a user pin and ID equal to 3296 and 21477 respectively.

3.3.3 Keypad test

Due to the fact that we want to verify users the is need for us to design a interface for the user to enter their credentials for this there is going to be use of a keypad for input and lcd for display.

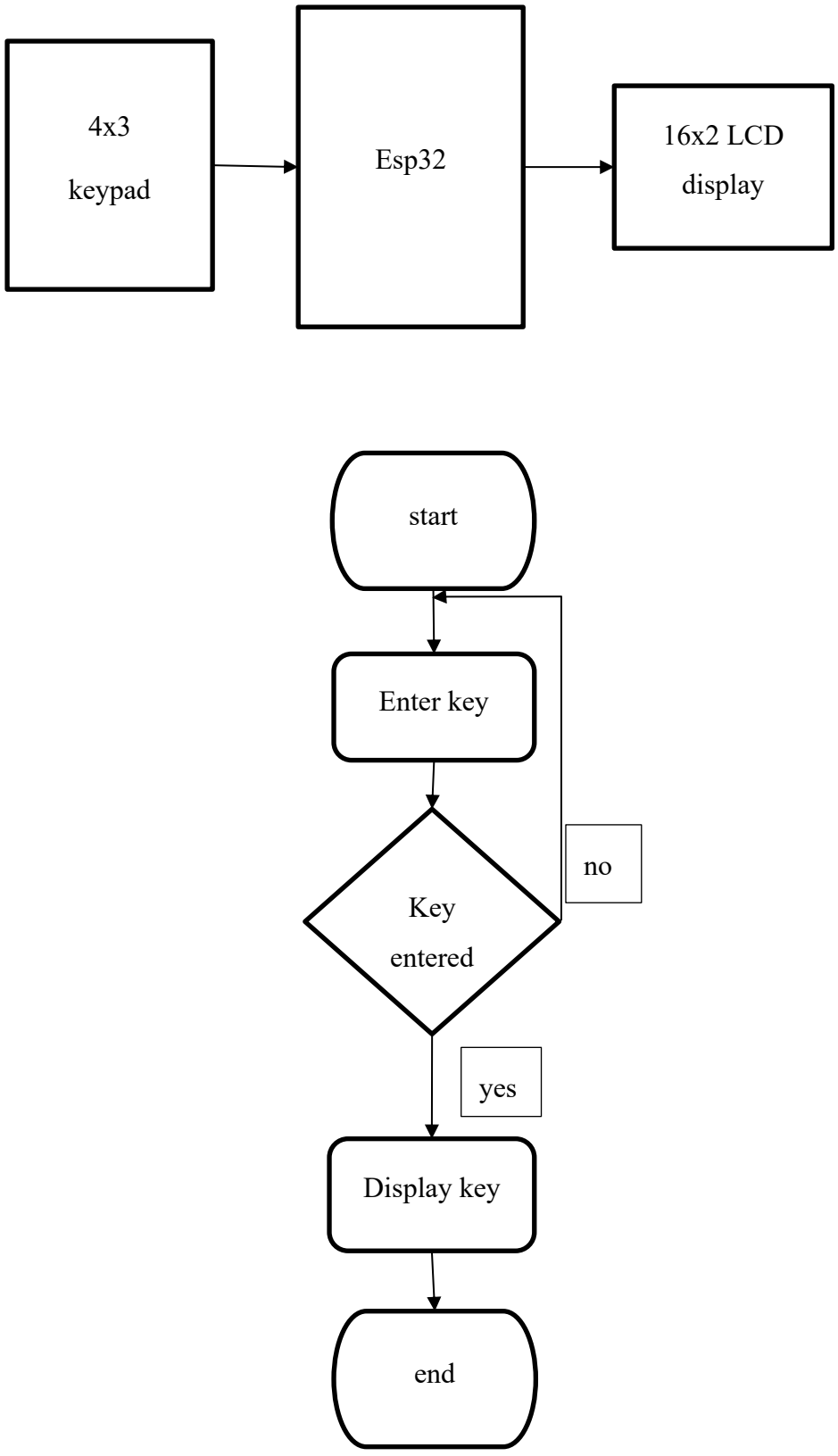


Figure 3.6: block diagram and flowchart of key test

Figure 3.5 shows the block diagram and the flow chart for the key test whereby the user is prompted to enter a key via the keypad and then the program will be displayed on the LCD. Below is a code to display that.

```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>


LiquidCrystal_I2C lcd(0x27, 16, 2);


const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
//row and pin connections to the keypad
byte rowPins[ROWS] = {16, 17, 5, 18};
byte colPins[COLS] = {19, 26, 32};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);//mapping keys to the pins on the esp32


void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("Press a key:");// prompting user for key
}
```

```
void loop() {
  char key = keypad.getKey();//storing key in variable key
  if (key) {
    lcd.clear();
    lcd.print("Key: ");//displaying key did LCD
    lcd.print(key);
  }
}
```

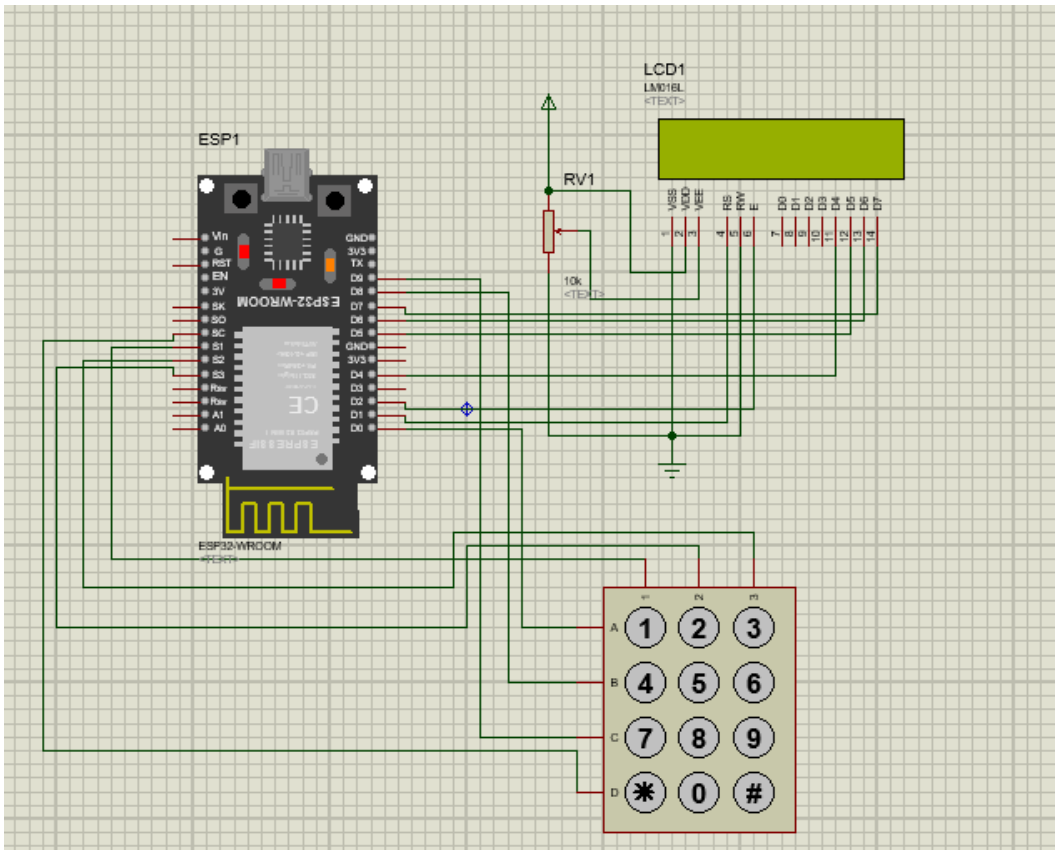


Figure 3.7: circuit for testing keypad

Figure 3.6 shows the circuit diagram of the keypad test whereby for each key pressed it will print it out on the LCD.

3.3.4 User verification testing

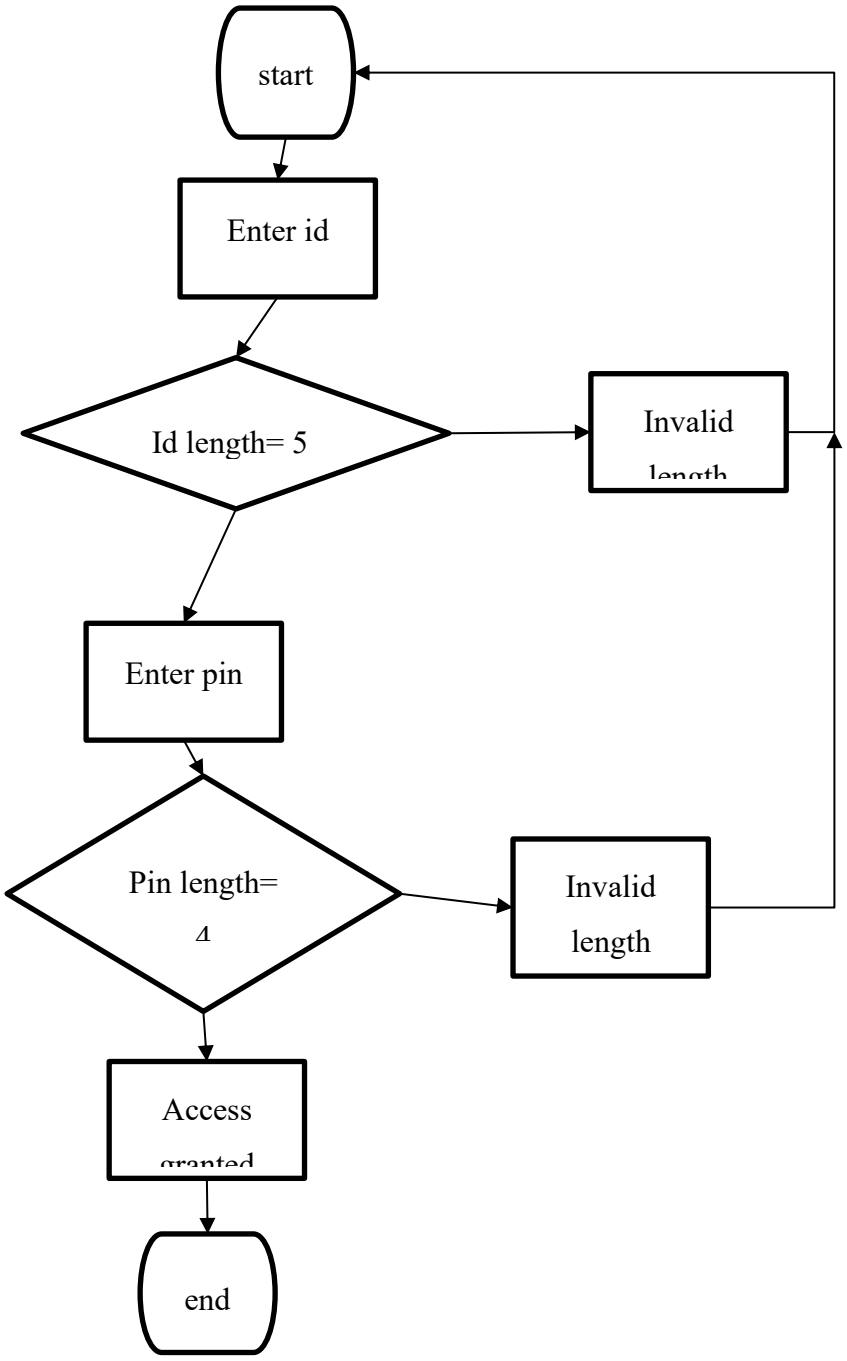


Figure 3.8:flowchart for user verification

Figure 3.7 shows the flow chart for the user verification process where it prompts you to enter a pin and ID and checks if the pin is equal to 4 digits and if the ID is equal to 5 digits and if they match those in the database it will display a success message. Below is the code for such a program

#include <LiquidCrystal.h>

```
#include <Keypad.h>

LiquidCrystal lcd(1, 2, 4, 5, 6, 7);// defining lcd pins

String storedID = "1234";
String storedPIN = "5678";

const byte ROWS = 4, COLS = 3;
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte rowPins[ROWS] = {0, 8, 9, 10};
byte colPins[COLS] = {11, 16, 15};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);// mapping of keys to pins

String enteredID = "";
String enteredPIN = "";
bool enteringID = true;

void setup() {
```

```
  lcd.begin(16, 2);
  lcd.print("Enter ID:");
}

void loop() {
  char key = keypad.getKey();
  if (!key) return;

  if (enteringID) {
    if (isdigit(key) && enteredID.length() < 4) {// for the simulation we will use id length as 4
      enteredID += key;
      lcd.setCursor(0, 1);
      lcd.print(enteredID);
    } else if (key == '#') {
      if (enteredID.length() == 4) {
        enteringID = false;
        lcd.clear();
        lcd.print("Enter PIN:");
      } else {
        lcd.clear();
        lcd.print("ID must be 4d");
        delay(1500);
        resetInputs();
      }
    }
  } else {
```

```
    if (isdigit(key) && enteredPIN.length() < 4) {
      enteredPIN += key;
      lcd.setCursor(0, 1);
      for (int i = 0; i < enteredPIN.length(); i++) lcd.print("*");
    } else if (key == '#') {
      lcd.clear();
      if (enteredID == storedID && enteredPIN == storedPIN) {// condition checking for id and
pin
        lcd.print("Access Granted");
      } else {
        lcd.print("Access Denied");
      }
      delay(2000);
      resetInputs();
    }
  }
}
// reset for next entry
void resetInputs() {
  enteredID = "";
  enteredPIN = "";
  enteringID = true;
  lcd.clear();
  lcd.print("Enter ID:");
```

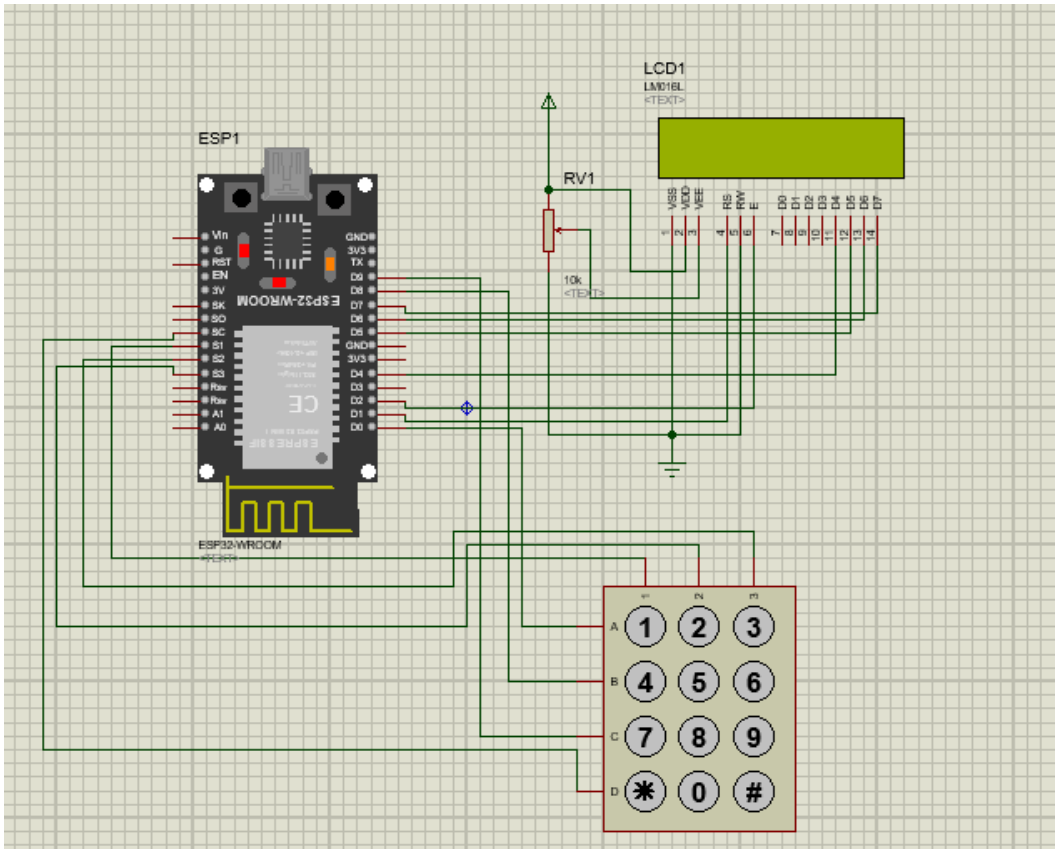


Figure 3.9: circuit diagram for user validation

Figure 3.8 shows the circuit diagram for user verification which will encompass entering pin and ID which will be compared to both registered in our database after it is connected to the Wi-Fi.

3.3.5 Ultrasonic sensor test

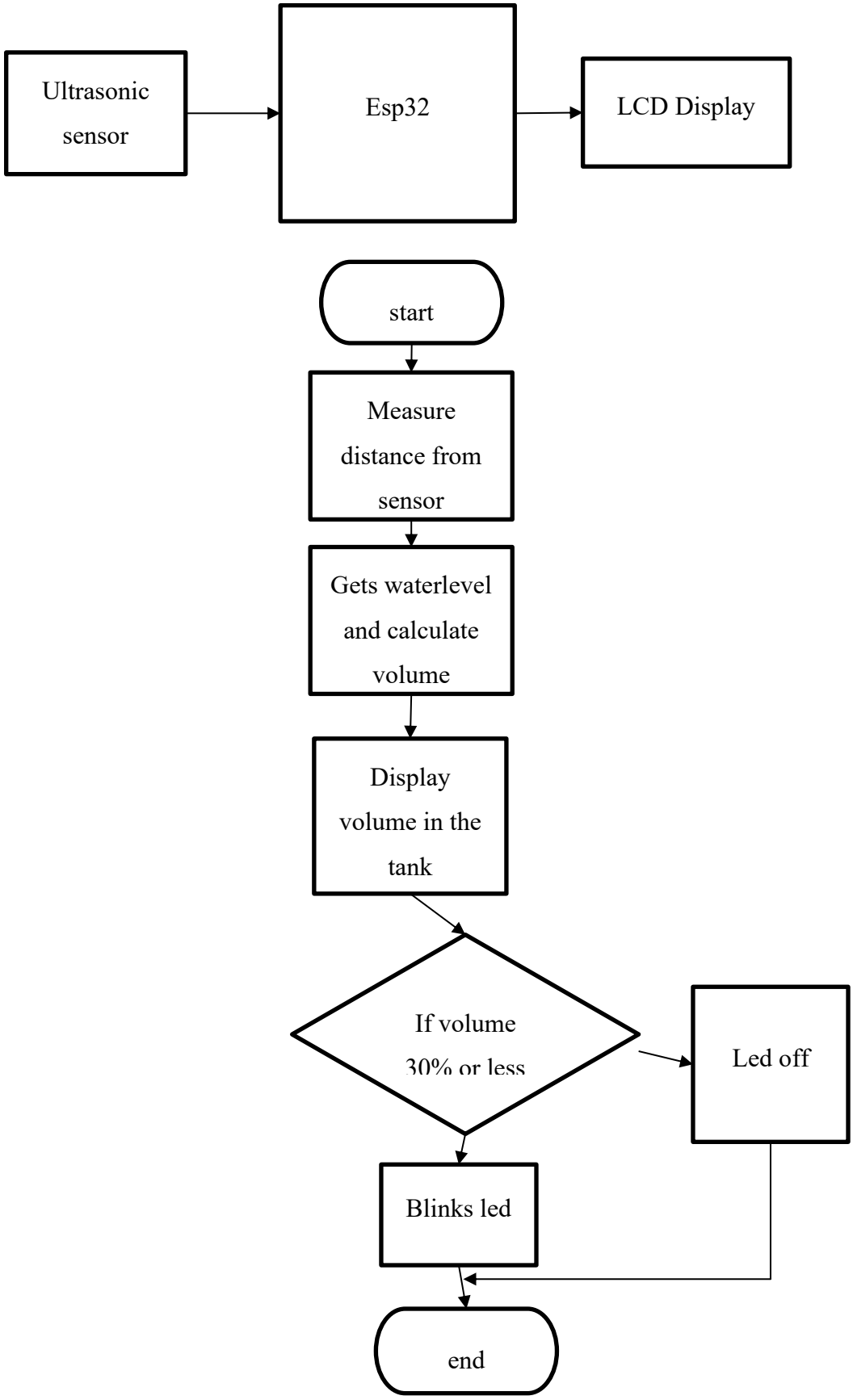


Figure 3.10:block diagram and flow chart for the ultrasonic test

Figure 3.10 shows the block diagram in flow chart of the ultrasonic sensor test whereby the whereby the ultrasonic sensor calculates the distance the water level by subtracting the height of the tank minus the distance it would have gotten and then uses a mathematical calculation calculate the volume and then display it on the LCD and if it less than or equal to 30% blinks the led.

```
#include <LiquidCrystal.h>


#define RS 1

#define E 2

#define D4 4

#define D5 5

#define D6 6

#define D7 7

LiquidCrystal lcd(RS, E, D4, D5, D6, D7);

#define TRIG_PIN 12

#define ECHO_PIN 13

#define LED_PIN 16 // Define LED pin

long duration;

float tankheight = 100;

float r = 5.642;

float pi = 3.14159265358;

float distance;

float dis;

float volume;

int percentage;


void setup() {
```

```
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT); // Set LED pin as output
  lcd.begin(16, 2);
  lcd.print("Ultrasonic Test");
  delay(1000);
  lcd.clear();
}
void loop() {
  // sending pulse ultrasonic sensor and then receiving it
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = duration * 0.0343 / 2;
  dis = tankheight - distance; // getting water level
  percentage = map(dis,95.60,-1016.07,0,100);//  // mapping water level to percentage
  volume = ( percentage/ tankheight) * (r * r * pi * tankheight); // calculation of volume
  // Check if percentage is 30% or below
  if (percentage <= 30) {
    digitalWrite(LED_PIN, HIGH); // Turn on LED
  } else {
    digitalWrite(LED_PIN, LOW); // Turn off LED
  }
```

```
  lcd.setCursor(0, 0);
  lcd.print("RML :");
  lcd.setCursor(4, 0);
  lcd.print(percentage);
  lcd.setCursor(0, 1);
  lcd.print("vol:");
  lcd.setCursor(4, 1);
  lcd.print(volume);
  delay(1000);
}
```

Above is the code of how to check the level in the tank by sending a beam using the trig pin and the catching the beam using the echo pin and then calculating the time it takes then calculating the distance. The we will calculate the volume having the assumption that our reserve tank is cylindrical in nature with a volume 10 000 litres with a height of 100m and radius. The formula for the distance from ultrasonic then level of fuel and finally volume is shown then

For calculating our fuel level, initial calculate the height of the fuel using the below formula

$$H = \frac{T \times S}{2}$$

Where H= height

T=time

S=speed of sound≈ 343 m/s → 0.03 43 cm/µs

$$V = \Pi R^2 H$$

Where R=radius

H= height

Resubstituting we get $R = \sqrt{\frac{V}{\Pi \times \mathrm{H}}}$

$$R = \sqrt{\frac{10000}{\Pi \times H}}$$

R=5.6418895835m

So for this demonstration we will take R as 5.64m. therefore our new volume becomes

$$V = \Pi \times 5.642^2 \times 100$$

V=10000.36926litres

Since our parameters are in meters and our ultrasonic sensor can only measure up to 2 meters only, so there is need for calibration scale using the table below where each centimetre represents a metre.

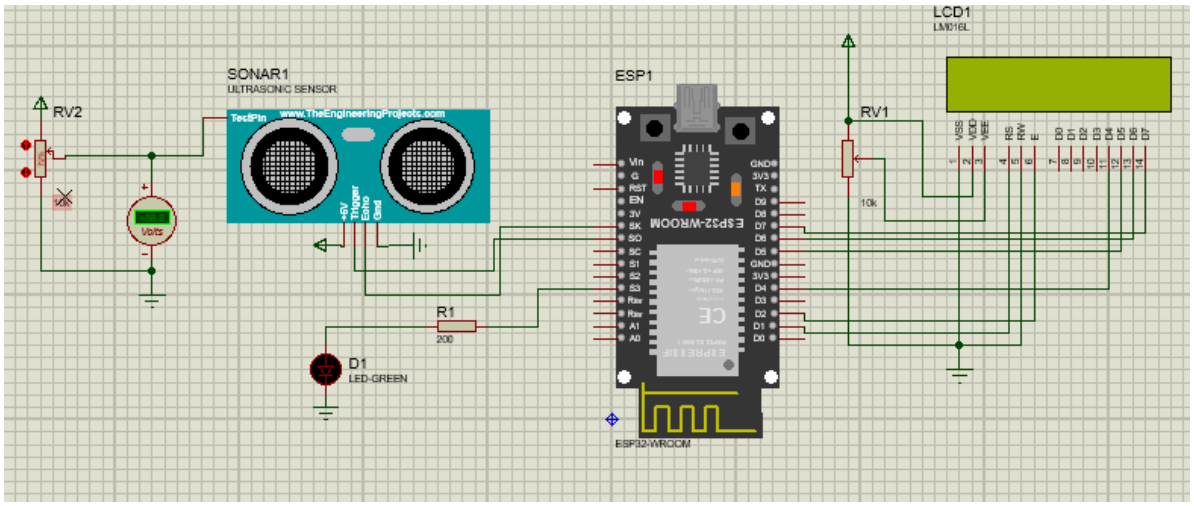


Figure 3.11: simulation circuit diagram for ultrasonic sensor

Figure 3.11 show the simulation circuit diagram for the ultrasonic sensor where it uses an ultrasonic sensor to measure the voltage across of a variable resistor and then uses it as input to the esp32. By varying the resistance of the variable resistor, the voltage changes proportionally thereby getting different values of volume.

Table 3.1: Calibration table for volume

Sensor Reading (cm)	Fuel Height (m)	Remaining Liters	Used Liters
1	99	9 900.365.57	10 000.369

12	88	8 800.32495	1 200.04431
35	65	6 500.240.02	3 500.12924
55	45	4 500.16617	5 500.20309
67	33	3 300.12186	6 700.24740
89	11	1 100.04062	8 900.32864
100	0	0	0

Table 3.2 shows the relationship between the reading measured from the ultrasonic sensor to the remaining reading that will be in the tank as compared to the actual reading that is in the tank on the physical phenomena. And then the actual number of litres that actual reading represents. It uses a scale of 1 cm as to 1 meter. Sensor reading the actual distance from the ultrasonic, fuel height is the difference between tank height and sensor reading multiplied by a 100. Remaining volume is the volume of the fuel based on fuel height and used liters is the difference between the tank's maximum capacity and the capacity at that instance.

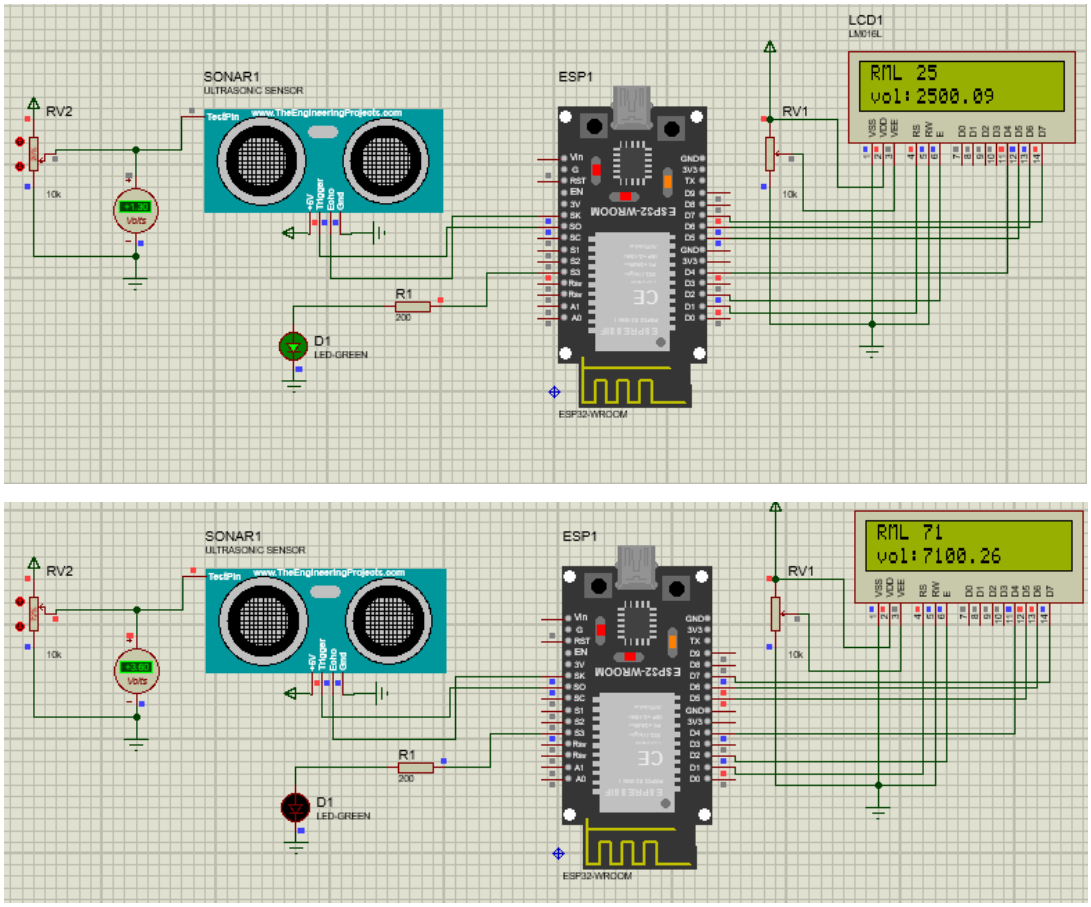


Figure 3.12: simulations of 2 different scenarios

Figure 3.12 shows simulation of the ultrasonic sensor in two different scenarios the first one being where RML(Remaining liters percentage) being 25 meaning the tank will be at 25 percent of its maximum capacity and since it is less than 30% it will turn the led to indicate that the fuel is running low. The second scenario is whereby the RML is 71 meaning 71% full and the led will be off.

3.3.6 Dispensing test

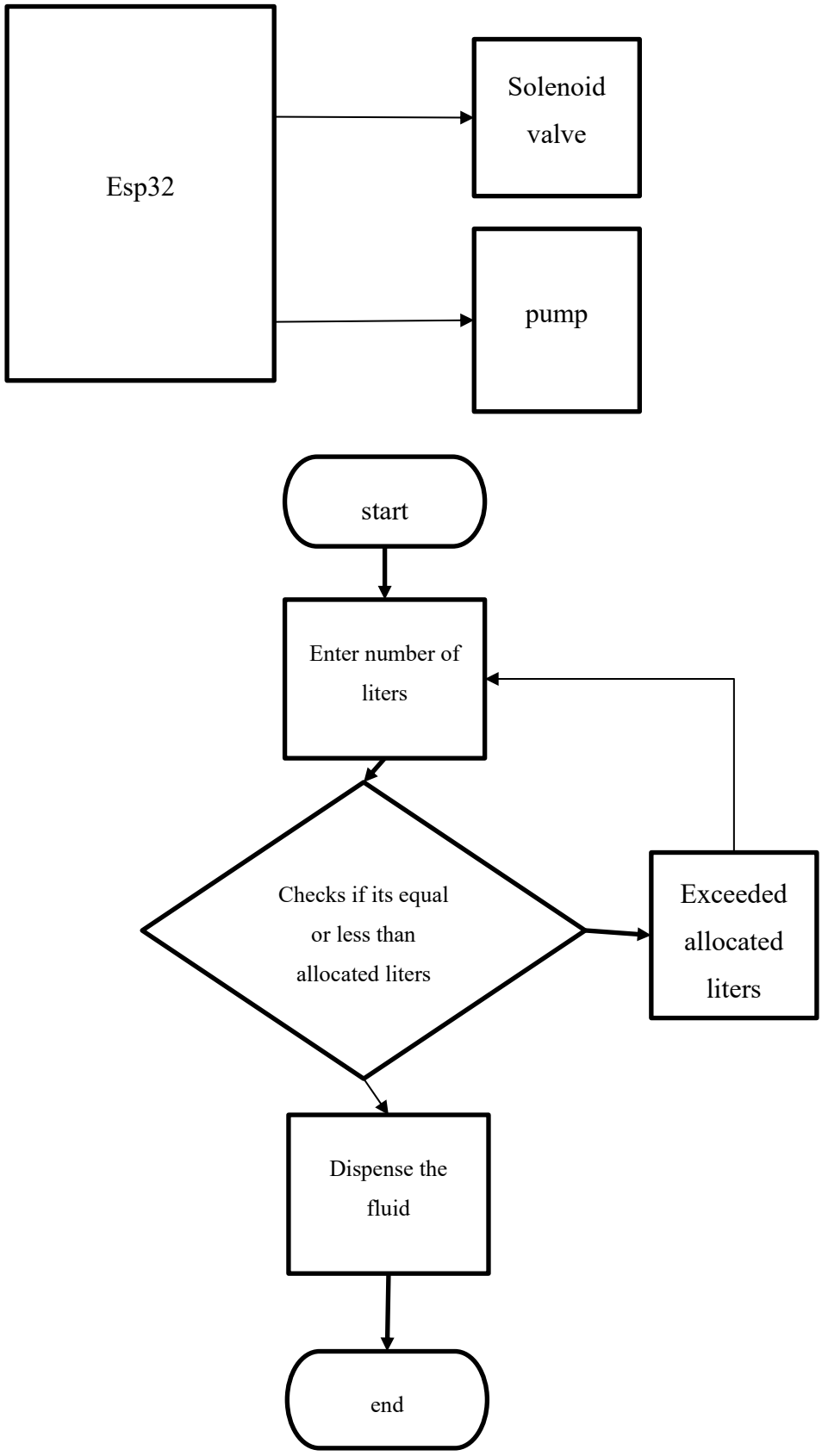


Figure 3.13: block diagram and flowchart for the dispensing system

Figure 3.13 shows the The block diagram and flowchart for their dispensing system whereby the user enters the desired number of liters and then the system we will calculate the time needed for it to dispense the fluid and whether its equal or less than the allocated number of liter they are allocated, if the entered number is equal or less than allocated liters it will dispense the fuel and if entered number of liters exceeds these allocated the system will display exceeded allocated number of liters, for demonstration purposes we are going to be using a led to represent the pump and solenoid valve and it would turn for the number of seconds that are required to for a successful dispense process of the entered number of liters. The code for the dispensing is given below.

```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0', '#'},
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
const int maxLiters = 10;// user allocated liters
const int ledPin = 13;
int enteredLiters = 0;

void setup() {
  lcd.begin(16,2);// initialize lcd
  pinMode(ledPin, OUTPUT);// setting pin 17 as output
  lcd.print("Enter liters:");
}

void loop() {
  char key = keypad.getKey();

  if (key != NO_KEY) {
    if (key == '#') {
      lcd.clear();
      lcd.print("Entered: ");
      lcd.print(enteredLiters);
      checkLiters();// calling checking function
      enteredLiters = 0;
      lcd.clear();
      lcd.print("Enter liters:");
    } else if (key >= '0' && key <= '9') {
      enteredLiters = enteredLiters * 10 + (key - '0');
      lcd.clear();
      lcd.print("Input: ");
```

```
      lcd.print(enteredLiters);
    }
  }
}

void checkLiters() {
// checking if enter liters is equal or less than prescribed liters
  if (enteredLiters <= maxLiters) {
    lcd.clear();
    lcd.print("Dispensing...");
    int ledTime = map(enteredLiters, 0, maxLiters, 0, 50);
    int countdown = ledTime;
    unsigned long startTime = millis();
    digitalWrite(ledPin, HIGH);
    while (millis() - startTime < ledTime * 1000) {// for the duration of dispensing have a clock countdown
      lcd.setCursor(0, 1);
      lcd.print("Time left: ");
      lcd.print(countdown);
      lcd.print("s");
      delay(1000);
      countdown--;
    }
    digitalWrite(ledPin, LOW);
  } else {
    lcd.clear();
```

```
    lcd.print("Exceeds maxLiters.");
    delay(2000);
  }
}
```

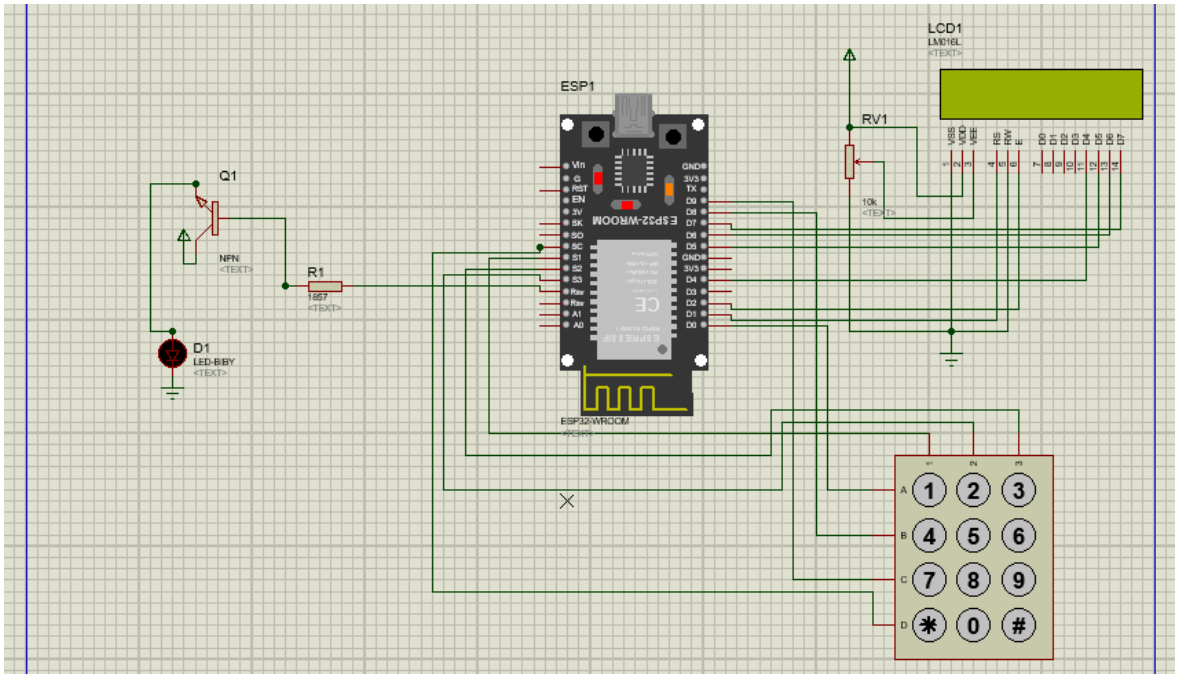


Figure 3.14: Simulation circuit for dispensing

Figure 3.14 shows the simulation circuit for the dispensing of fluid where we are connecting LCD for displaying information keypad for entering number of litters ESP 32 for doing the logical calculations and then the dispensing side of this is a resistor connected to the base of the transistor then the led.

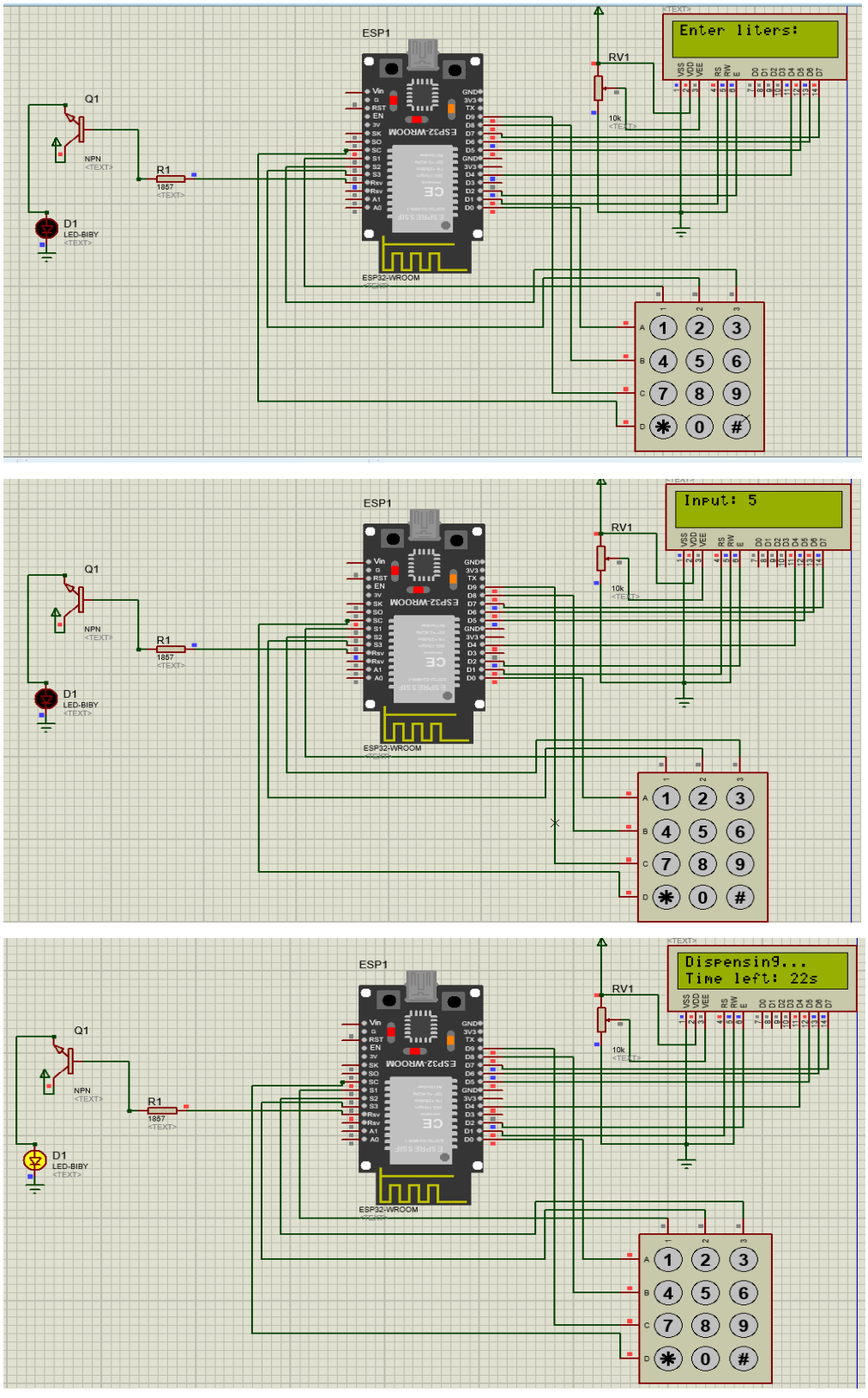


Figure 3.15:a successful dispense

Figure 3.15 shows a successful dispense whereby the user is prompted to enter number of liters they want, and the user enter 5 liters then the system doing a countdown as the led will be on as a representation of the dispensing process.

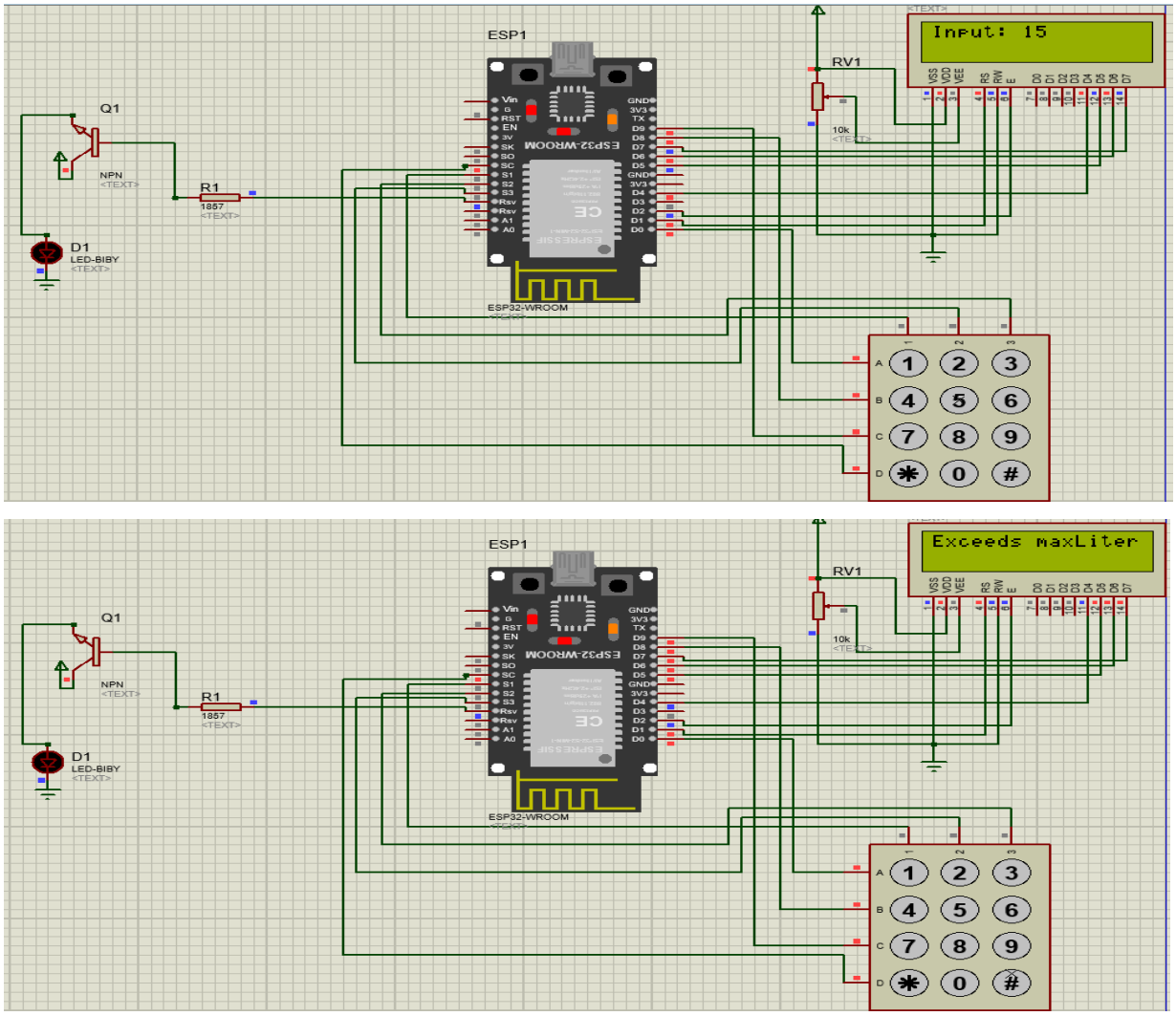


Figure 3.16: a failed dispense

Figure 3.16 shows a failed dispensing whereby the user enters 15 liter which is exceeds the allocated 10 liters and the user getting a error message that their entered liters is more than the ones prescribed to them.

Calculation

Since we cannot interface the ESP 32 to the relay switch directly, we had to use a transistor so that we could step up the voltage so that it can be able to activate the relay So the calculation into the resistance needed for the resistor is as follows

$$Ib = \frac{Ic}{hfe} = \frac{70mA}{100} = 0.7mA$$

Where Ib=base current

Ic=collector current

hfe= current gain of BC547

Doubling Ib makes it fully saturated so therefore we must find a resistor value to do that

$$Vrb = \frac{Vesp - Vbe}{Ib \times 2} = \frac{3.3 - 0.7}{0.0007 \times 2} = 1857\Omega$$

Where Vesp =esp32 gpio voltage

Vrb=voltage across base resistor

Vbe= voltage drop across base and emitter

So for the base resistor any value between 1K ohm up to 2K ohms is ok for saturating the transistor

This approach is used since it reduces the amount of current being drawn by the relay from the microcontroller than if it was connected directly and also making it easier to saturate the transistor

3.4 Conclusion

This chapter described the design used to develop an automated, priority-based fuel procurement system. By combining various sensors, a microcontroller, and a central database, the system eliminates manual inefficiencies and introduces transparency, real-time monitoring, and intelligent fuel management aligned with organizational priorities.

4 CHAPTER 4: SYSTEM IMPLEMENTATION

4.1 Introduction

This chapter presents the implementation details of the system. The chapter discusses how the system was implemented both on the hardware side (sensors, actuators) and the software side (microcontroller code and backend scripts). The chapter discusses how user authentication, water level sensing, dispensing control, and database updates are integrated to achieve an integrated solution. In this chapter we are dealing with the implementation of the functions of the project this encompasses functions like data retrieval from the database user validation testing or fuel in the tank and dispensing and the fuel.

4.2 System functions result

This part of the implementation we are testing the functions that we described in the design chapter 3 and then putting out the results of those tests. This chapter encompasses both the software simulations and the real-life tests of the functionalities described in chapter 3 and also how they will implement the objectives ff this system.

4.2.1 Database structure and communication with esp32

The database design and implementation for the Fuel Dispensing System involves several key processes, beginning from requirements gathering, through the design of the database schema, and culminating in physical implementation and integration with the ESP32 microcontroller via PHP scripts. The primary aim of the database is to manage user authentication, store the amount of fuel each user is entitled to, and log each dispensing transaction for accountability and traceability.

The system is designed to manage the registration and authentication of users using a unique ID and PIN, storage of the user's fuel balance in liters, and logging every fuel dispensing transaction, including the user ID, dispensed amount, remaining balance, and timestamp. From this analysis, two main entities were defined: a table for storing user credentials and balances, and a log table to record individual fuel dispensing events. Each user can perform multiple dispensing operations, establishing a one-to-many relationship between users and dispense logs.

Requirement Analysis

The primary purpose of the system's database is to manage:

- ❖ User registration and authentication using a unique ID and PIN.
- ❖ Storage of each user's fuel balance (liters remaining).
- ❖ Logging each fuel dispensing transaction, including:
  - User who dispensed fuel
  - Amount dispensed
  - Balance after dispensing
  - Timestamp of the transaction

4.3 Conceptual Database Design

From the analysis, the following two main entities were identified:

User

- ❖ Represents an individual registered in the system.

Dispense Log

- ❖ Represents each transaction where fuel is dispensed.

Each user may perform multiple dispensing transactions, resulting in a one-to-many relationship between users and dispense logs.

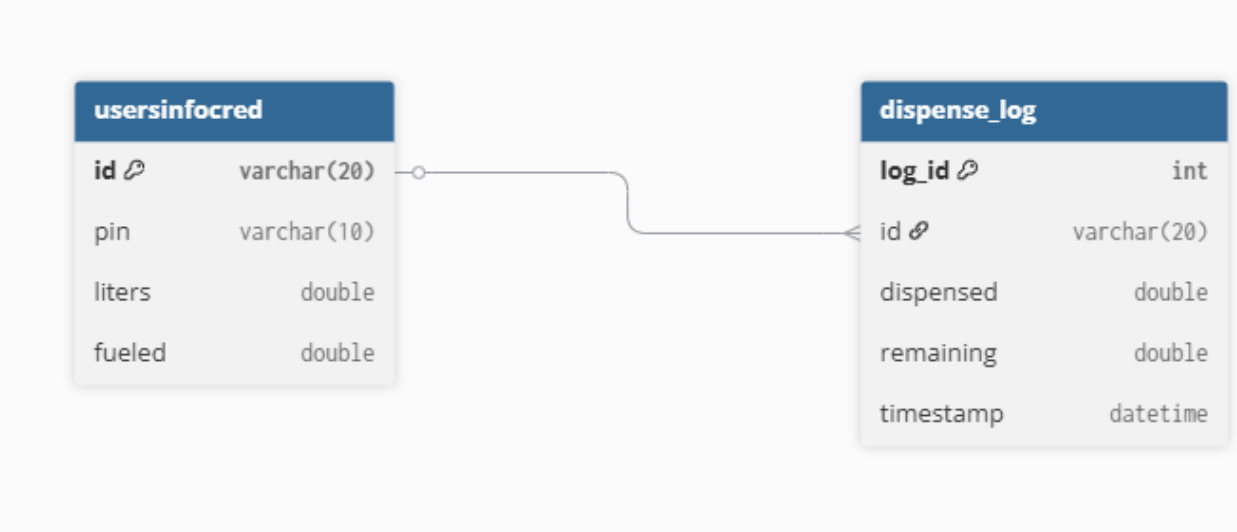


Figure 4.1: Entity-Relationship Diagram

The Entity-Relationship Diagram (ERD) for the system consists of a table called usersinfocred, which holds fields for user ID, PIN, liters remaining, and total fueled liters. Linked to this is the dispense_log table, which records the unique log entry ID, the user ID as a foreign key, the amount dispensed in a particular transaction, the remaining liters after the transaction, and the timestamp of the dispensing event. The relationship ensures that for every dispensing transaction, the system retains a historical record tied to the specific user. Figure 4.1 shows the conceptual ERD of the system.

The logical design of the tables is as follows. The usersinfocred table contains the fields: id as a VARCHAR(20) serving as the primary key, pin as a VARCHAR(10) which is unique to prevent duplicate entries, liters as a DOUBLE representing the user's current fuel balance, and fueled as a DOUBLE representing the total fuel already dispensed by the user. The dispense_log table includes log_id as an INT serving as the primary key with auto-increment, id as a VARCHAR(20) acting as a foreign key linked to usersinfocred, dispensed as a DOUBLE indicating the volume of fuel dispensed during the transaction, remaining as a DOUBLE reflecting the fuel left after dispensing, and timestamp as a DATETIME indicating when the transaction occurred.

The physical implementation uses MySQL. The database named usersinfo is created, followed by the two tables. The SQL commands to implement this design are as follows:

CREATE DATABASE usersinfo;

USE usersinfo;

```
CREATE TABLE usersinfocred (
    id VARCHAR(20) NOT NULL PRIMARY KEY,
    pin VARCHAR(10) UNIQUE,
    liters DOUBLE DEFAULT 0,
    fueled DOUBLE DEFAULT 0
);

CREATE TABLE dispense_log (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    id VARCHAR(20),
    dispensed DOUBLE,
    remaining DOUBLE,
    timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id) REFERENCES usersinfocred(id)
);
```

For inserting initial data into the users table is:

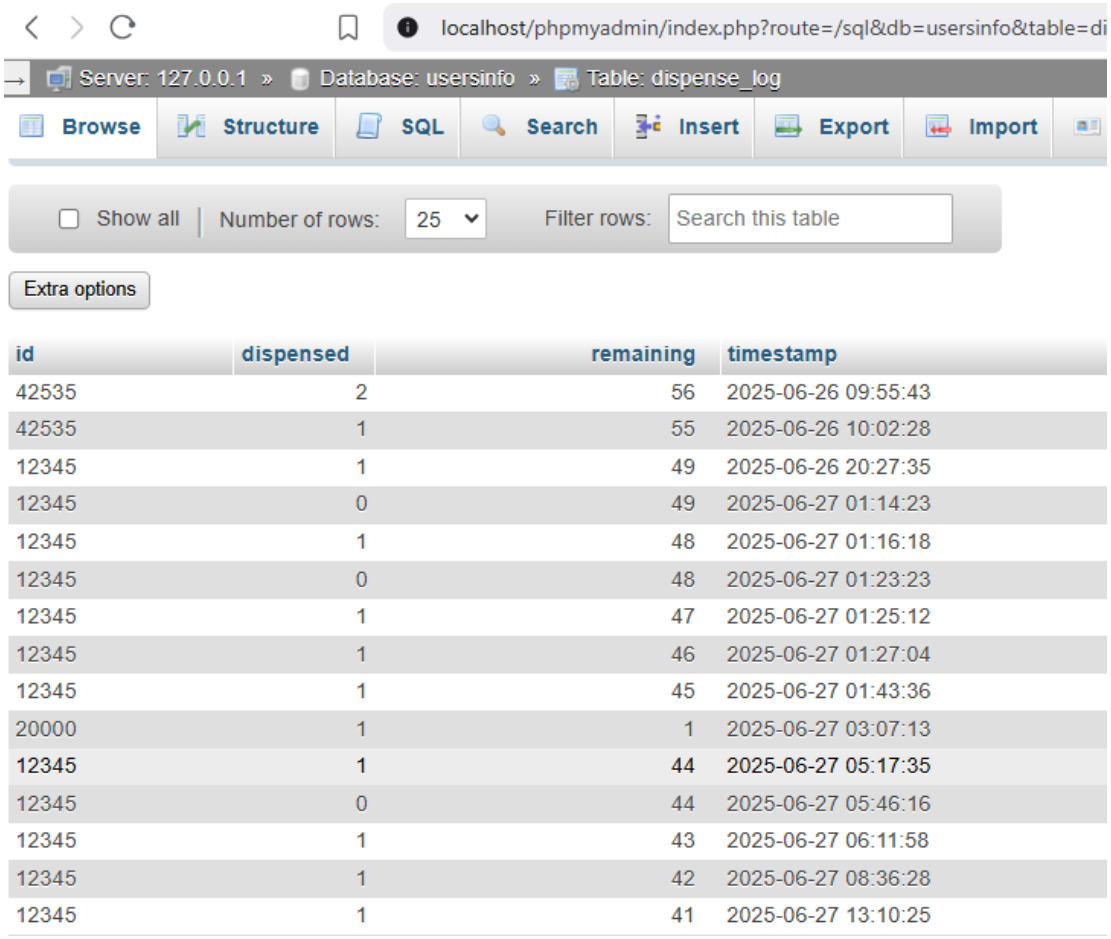
```
INSERT INTO usersinfocred (id, pin, liters, fueled)
VALUES ('43930', '1234', 50.0, 0);
```

Extra options

	id 1	pin	liters	fueled
Edit Copy Delete	12345	1111	42	2
Edit Copy Delete	20000	2222	1	1
Edit Copy Delete	21477	3296	48	NULL
Edit Copy Delete	22364	2332	57	NULL
Edit Copy Delete	23456	6666	400	NULL
Edit Copy Delete	24673	7842	15	NULL
Edit Copy Delete	25005	4356	48	NULL
Edit Copy Delete	25802	4356	26	NULL
Edit Copy Delete	32356	4561	5	NULL
Edit Copy Delete	34567	3333	150	NULL
Edit Copy Delete	36964	2423	122	NULL
Edit Copy Delete	42535	4812	55	1
Edit Copy Delete	45678	5555	300	NULL
Edit Copy Delete	46321	854	76	NULL
Edit Copy Delete	55975	2426	39	NULL

Figure 4.2: usersinfocred table viewed on the localhost

Figure 4.2 shows the userinfocred table viewed on the localhost where each user has properties ID which identifies them , pin for verification, liters which is the fuel allocated to them and fueled which is the number of liters the user would have dispensed.



id	dispensed	remaining	timestamp
42535	2	56	2025-06-26 09:55:43
42535	1	55	2025-06-26 10:02:28
12345	1	49	2025-06-26 20:27:35
12345	0	49	2025-06-27 01:14:23
12345	1	48	2025-06-27 01:16:18
12345	0	48	2025-06-27 01:23:23
12345	1	47	2025-06-27 01:25:12
12345	1	46	2025-06-27 01:27:04
12345	1	45	2025-06-27 01:43:36
20000	1	1	2025-06-27 03:07:13
12345	1	44	2025-06-27 05:17:35
12345	0	44	2025-06-27 05:46:16
12345	1	43	2025-06-27 06:11:58
12345	1	42	2025-06-27 08:36:28
12345	1	41	2025-06-27 13:10:25

Figure 4.3:dispense_log table viewed on the localhost

Figure 4.3 shows the dispense_log table viewed on the localhost where id is used to identify the user dispensed being the amount of fuel dispensed by the user of the corresponding id , remaining being the number of liter remaining after the dispense and timestamp being the time and date which that dispense would have happened. Further more users only appear on this log if the fuel.

To integrate the database with the ESP32, two PHP scripts were developed and deployed on a local XAMPP server. The first script, verify_user.php, accepts the user ID and PIN sent from the ESP32 and verifies them against the database. If the credentials match, it returns the number of liters remaining for the user; otherwise, it returns an error message.

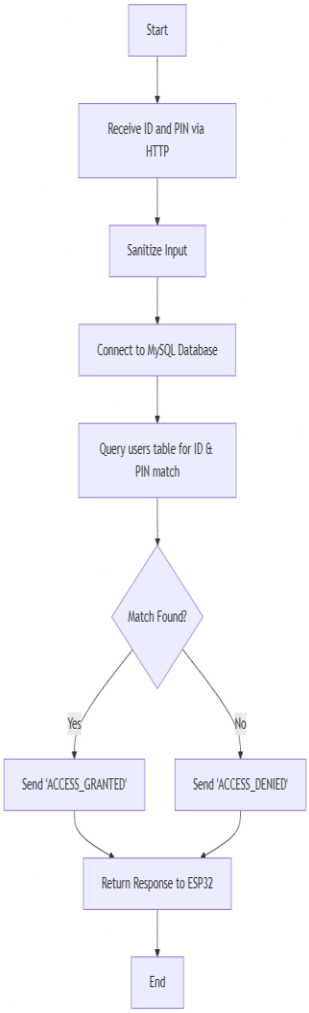


Figure 4.4:flowchart for user verification php file

Figure 4.4 shows the flow chart for user verification php file where the ESP32 sends an HTTP GET or POST request to the user's ID and PIN. The PHP script receives the request and cleanses the input to avoid SQL injection. The script queries the `usersinfocred` table to determine whether the provided credentials match a record within the database. In case of successful match, the script sends back a success code. In case of unsuccessful match, the script sends back a failure response. Below is the code for php file.

<?php

header('Content-Type: text/plain');

$host = \"localhost\";

$user = \"root\";

$pass = \"\";

```
$dbname = \"usersinfo\";

$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die(\"Connection failed: \" . $conn->connect_error);
}

$id = $_GET['id'] ?? '';
$pin = $_GET['pin'] ?? '';

$sql = \"SELECT liters FROM usersinfocred WHERE id = ? AND pin = ?\";
$stmt = $conn->prepare($sql);
$stmt->bind_param(\"ss\", $id, $pin);
$stmt->execute();
$result = $stmt->get_result();

if ($row = $result->fetch_assoc()) {
    echo $row['liters'];
} else {
    echo \"error\";
}

$stmt->close();
$conn->close();
?>
```

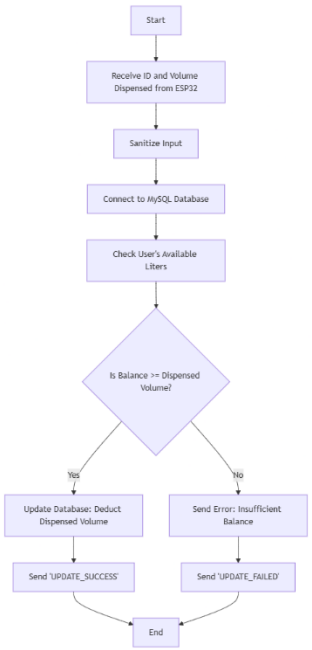


Figure 4.5: flowchart for the udate liters php file

Figure 4.5 shhows the flowchart for the udate liters php file where the localhost receives the user ID, the new liters balance, and the amount dispensed during the transaction. It updates the corresponding record in the usersinfocred table and logs the transaction into the dispense_log table to ensure a historical record is kept for auditing and reporting. The script is implemented as follows:

```
<?php
header('Content-Type: text/plain');

$host = \"localhost\";
$user = \"root\";
$pass = \"\";
$dbname = \"usersinfo\";

$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die(\"Connection failed: \" . $conn->connect_error);
```

```
}

$id = $_GET['id'] ?? '';
$liters = floatval($_GET['liters'] ?? '');
$fueled = floatval($_GET['fueled'] ?? '');

$sql = \"UPDATE usersinfocred SET liters = ?, fueled = ? WHERE id = ?\";
$stmt = $conn->prepare($sql);
$stmt->bind_param(\"dds\", $liters, $fueled, $id);
$updateSuccess = $stmt->execute();
$stmt->close();

$logSql = \"INSERT INTO dispense_log (id, dispensed, remaining) VALUES (?, ?, ?)\";
$logStmt = $conn->prepare($logSql);
$logStmt->bind_param(\"sdd\", $id, $fueled, $liters);
$logStmt->execute();
$logStmt->close();

echo $updateSuccess ? \"success\" : \"error: could not update\";
$conn->close();
?>
```

Through these processes, the system achieves a secure and efficient method of handling user authentication and fuel allocation while maintaining complete transaction logs for all dispensing activities. The integration of the ESP32 microcontroller with the PHP scripts and MySQL database ensures reliable data exchange and real-time updates, forming a robust foundation for the operation of the system.

4.2.2 User verification

User is a two-step process:

1.ID Entry Stage

2.PIN Entry Stage

Both processed by the `handleInput()` function.

ID Entry:

- Users enter a 5-digit ID.
- Confirmed with a press of `#`.

Short code for checking if id is 5 digits

```
if (key == '#') {// when When user presses the # key
  if (id.length() == 5) {//Checking if it's equal to 5 digits
    entryStage = PIN_ENTRY;//System proceeds to entry of ID
  } else {
    resetSystem("ID must be 5");//if ID is not equal to 5 digits it jumps to the reset system function and presses ID must be equal to 5 on the LCD
  }
}
```

PIN Entry

- Users enter 4-digit PIN.
- Confirmed when `#` is pressed.

Short code for checking if pin is 4 digits

```
if (key == '#') {    //When key pressed is equal to #
  if (pin.length() == 4) {    //checks if pin is equal to 4 digiits
    verifyUser(id, pin);//Taking pain and ID into the verify user function which verifies the user
  } else {
    resetSystem("PIN must be 4");   //If pin is not equal to 4 digits it goes to the reset function and print out pin must be 4
```

```
  }
}
```

The ESP 32 then concatenates pin and ID to the end point you are real Endpoint which creates PHP requests through the HTTPClient library.

ESP 32 sends this to our local database which will process the URL using the verify user PHP file in return a response if the ID and pin have corresponding and record in the database.

```
void verifyUser(String id, String pin) {
  String url = apiEndpoint + "?id=" + id + "&pin=" + pin; //Concatenating ID and pin to the URL string
  HTTPClient http;//creating a URL object
  http.begin(url);//defining the URL object which now has ID and pin added to it
  int httpCode = http.GET();//Sending URL to the local hosts

  if (httpCode > 0) {//checking for the response of the URL after it has been sent to the local hosts
    String response = http.getString();//if the response meet one of the criterias in the database that is ID and pin are existent in the database it prints out in the requests

if (response.startsWith("error")) {//if the URL did not meet any of the criterias in the database
      resetSystem("Invalid login");//It's jumped the resets function and prints out invalid login
    } else {
      int commaIndex = response.indexOf(',');
      allowedLiters = response.substring(0, commaIndex).toFloat();
      lcd.clear();
      lcd.print("Allowed: ");
      lcd.print(allowedLiters);
```

```
    lcd.setCursor(0, 1);
   }
lcd.print("Enter amt (L)#");
    entryStage = LITER_ENTRY;
   }
  } else {
   resetSystem("HTTP error");
  }
 http.end();
```

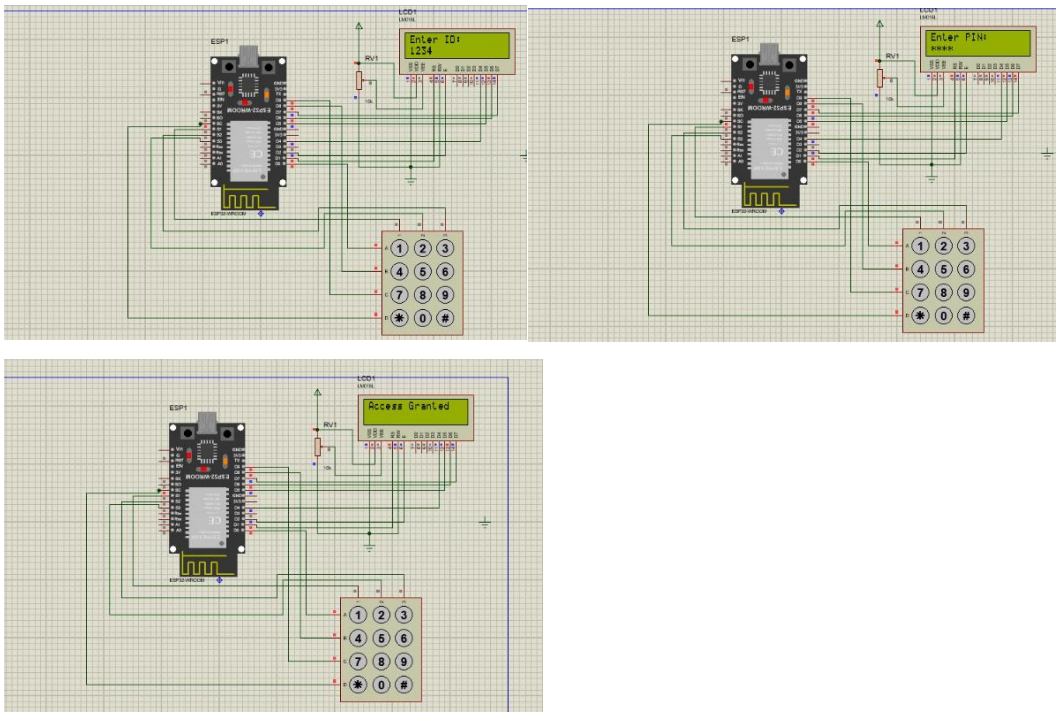


Figure 4.6:simulation of the verify use with keypad

Figure 4.1 shows the simulation of the verify user test with keypads tests we will be entering 4-digit pin and ID and if the length was less than 4(According to the code logic for the simulation we entered ID being 4 digits but when we do the real-life if ID should be to 5 digits) it was going to display an access denied message auto if it wasn't equal to the one that was stored in the microcontroller, for the simulation the user enters a ID which is “1234” and in “5678” but we will not see the pin as it puts out star to keep their ID confidential and the system will grant them access.

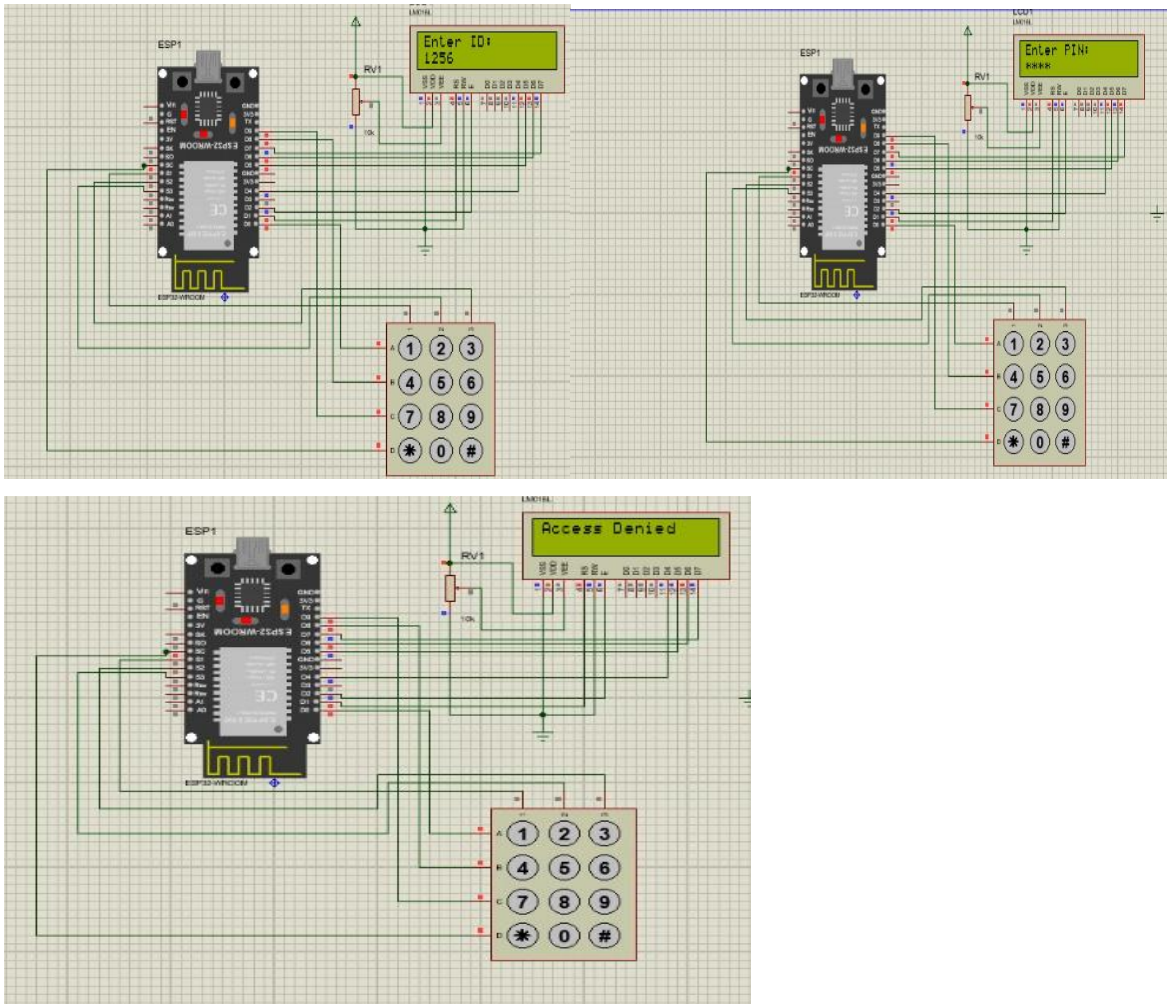


Figure 4.7:verify user when we enter wrong pin and ID

Figure 4.7 show the verify user test when we enter a wrong ID and pin then returns a access denied message.

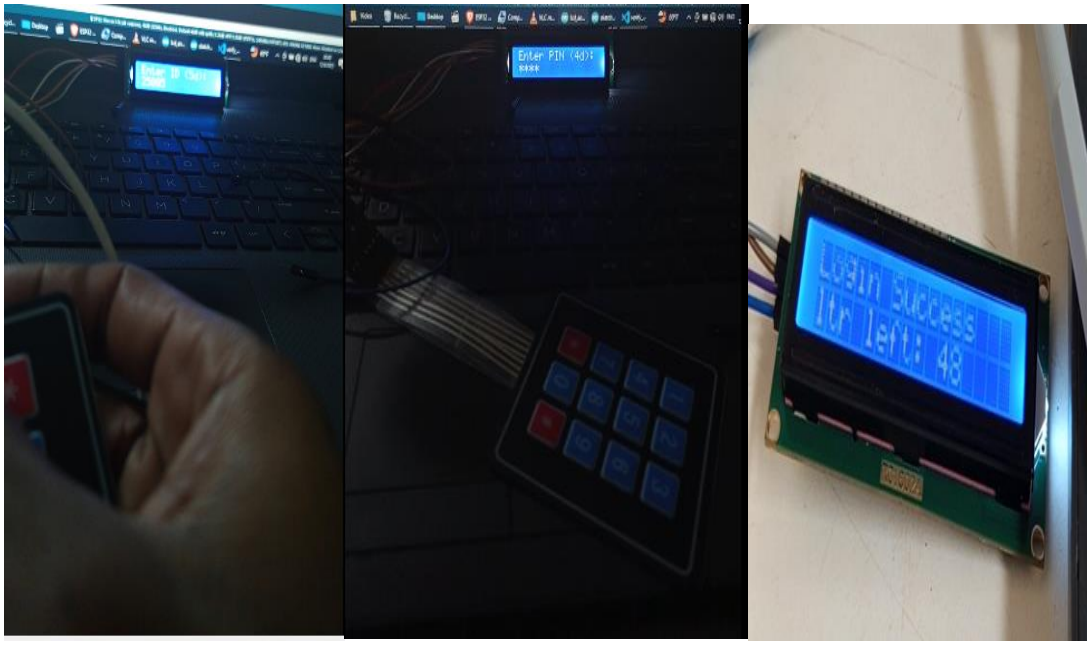
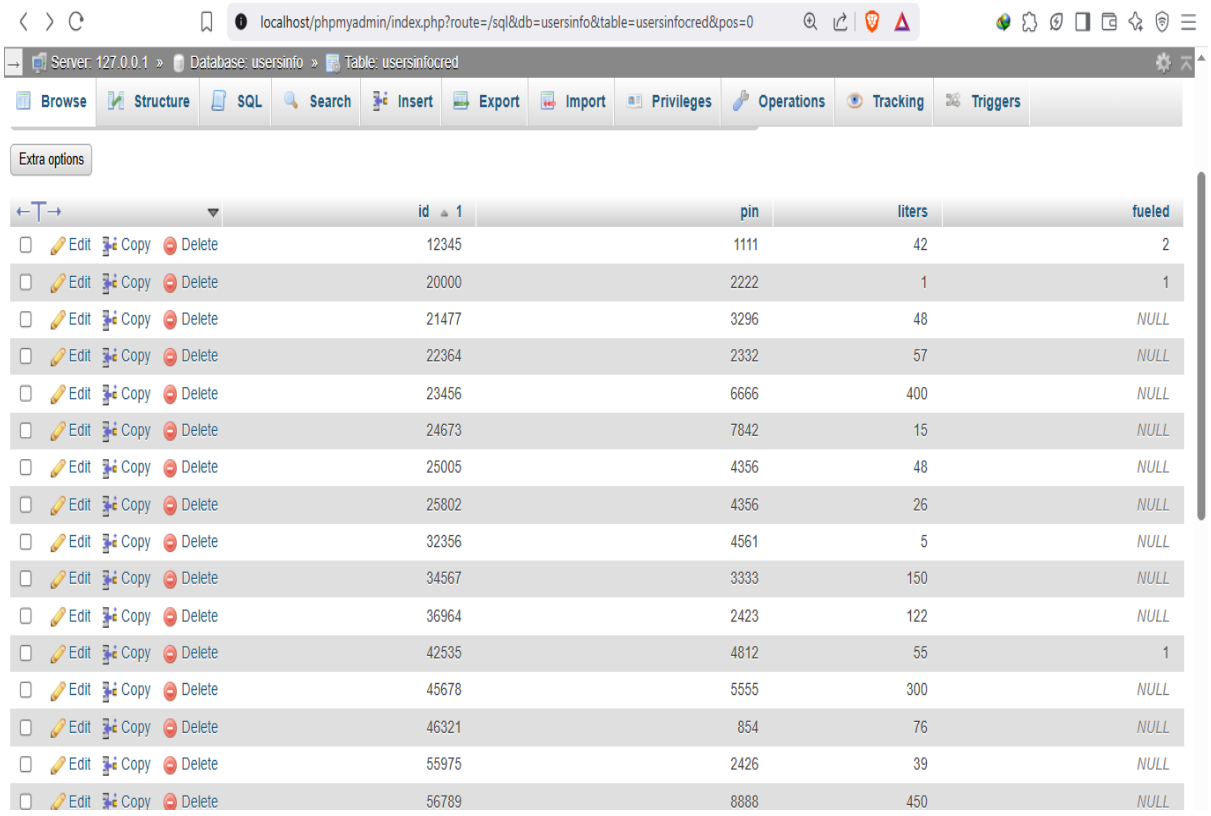


Figure 4.8: user verification in real life test

Figure 4.8 shows the user entering an ID “25005” then the user is prompted to enter a pin then the user enters a 4-digit pin “4356” the esp32 generates a http request to check whether such a user exits in the data which is sent to the localhost and finally the LCD printed “Login Success” meaning such a user exists in the database. The ltr left is the number of liters that the user can full at that point in time or less than that.



	id	pin	liters	fueled
Edit Copy Delete	12345	1111	42	2
Edit Copy Delete	20000	2222	1	1
Edit Copy Delete	21477	3296	48	NULL
Edit Copy Delete	22364	2332	57	NULL
Edit Copy Delete	23456	6666	400	NULL
Edit Copy Delete	24673	7842	15	NULL
Edit Copy Delete	25005	4356	48	NULL
Edit Copy Delete	25802	4356	26	NULL
Edit Copy Delete	32356	4561	5	NULL
Edit Copy Delete	34567	3333	150	NULL
Edit Copy Delete	36964	2423	122	NULL
Edit Copy Delete	42535	4812	55	1
Edit Copy Delete	45678	5555	300	NULL
Edit Copy Delete	46321	854	76	NULL
Edit Copy Delete	55975	2426	39	NULL
Edit Copy Delete	56789	8888	450	NULL

Figure 4.9:usersinfocred database

Figure 4.9 shows a usersinfocred table contain information of all the users each user having attributes id which is the user identification, pin for the corresponding user, liters which is the number of liters allocated by the system and fueled being the number of liters the user would have fueled. Also, as we can see on the 7th column the exists a user with id equal to 25005 and pin 4356 and fueled column having 48 as the lcd returned in figure 4.8.

4.2.3 Dispensing of the system

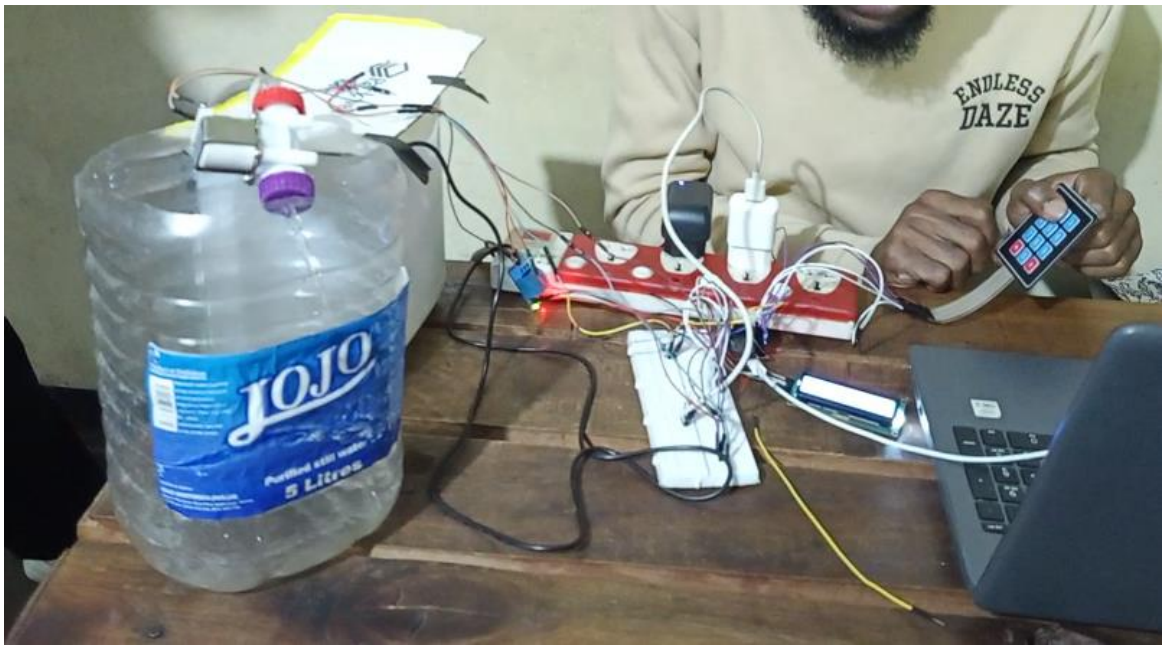


Figure 4.10:system dispensing

Figure 4.10 shows the system dispensing where the system first measures the volume in the tank and then dispenses the fluid using the pump and the solenoid valve combination and the ultrasonic sensor which will measure the distance and then calculate the volume in the tank before and after dispensing so that we can see whether it's measuring volume correctly. The system uses an ultrasonic sensor mounted on the top of the water tank to measure the level of water available.

Figure 4.11:ultrasonic sensor mounted on the tank(left) and volume calculated on display before dispensing

Figure 4.11 shows the ultrasonic sensor being mounted on the tank on the left where the ultrasonic sensor will distance to the water level which is interpreted as “Dist” on the figure on the right which is this case is 3.26 cm then subtract from tank height and will be represented as “LVl” on the figure on the right which in this case is 13.9 cm and the volume calculated and displaced as “VOL” (right) which in this case is 3.7 liters.

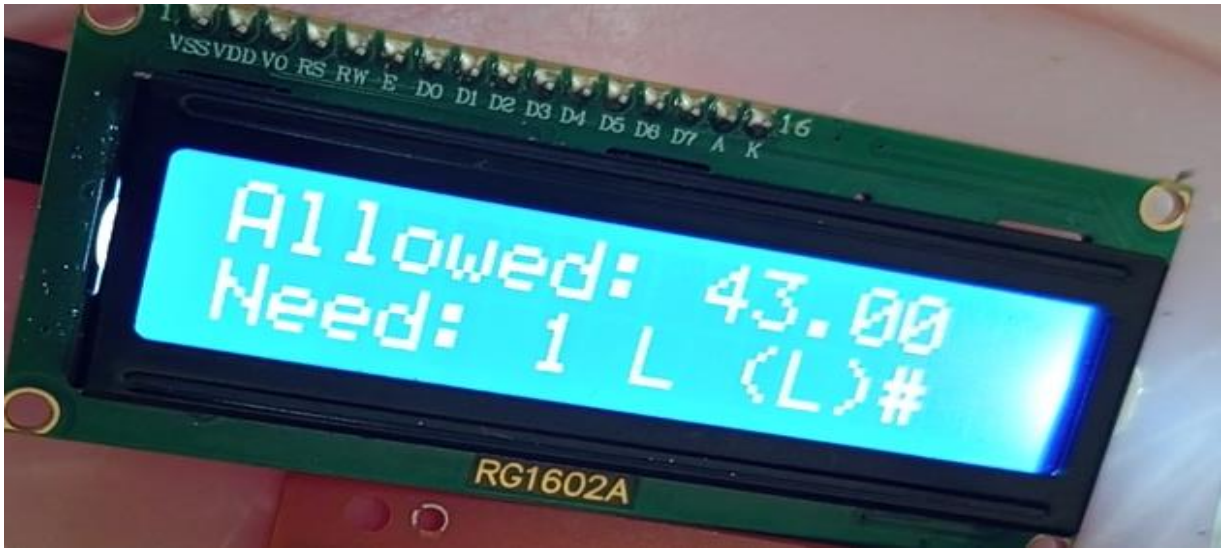


Figure 4.12: system prompting user to enter liters they need

Figure 4.12 shows the system showing the number of liters allocated to the user and is represented by “Allowed” in this case being 43 liters prompting the user to enter the number of liters they want to the fuel represented by “Need” in this case the user entered 1. The system checks whether the entered volume is less than or equal to the amount that users being allocated

and if so, it goes on to turn on the pump and it also calculates the amount of time needed to require to dispense the entered number of liters.

Figure 4.13: system dispensing

Figure 4.13 shows LCD displaying the countdown as the system is dispensing the 1 liter on the left and on the right the system is dispensing.

Figure 4.14: volume description of tank after filling 1 Liter

Figure 4.14 shows description of the tank after filling 1 liter since the user fuelled the “Dist” changed from 3.26cm to 7cm, “Lvl” changed from 13.9cm to 10.1cm and “VOL” changed from 3.7L to 2.7L meaning the fuel dispensing was one liter.

Figure 4.15: Tank status when volume is below 30% maximum.

Figure 4.15 shows the tank where “Dist” is the distance measured by the ultrasonic sensor in this case being 16.9 cm and LV1 being the height of fluid in the tank in this case being 0.2 cm, VOL being the volume in the tank in this case being 0.0 liters is is due to round up to 1 decimal place. Further since the tank capacity is now 0.0 liters which is lower than 30% of its maximum capacity(4.6L from computation of tank height(17.1cm) multiplied by breath (14.1) multipled by with tank width (19.1cm)), the led will be turned on as an alert to the user.

4.3 Conclusion

This chapter has presented the complete implementation features of the dispensing system. The hardware and software components were designed and integrated to offer a working and automated solution. Hardware-wise, components such as the ESP32 microcontroller, ultrasonic sensor, keypad, LCD display, relay module, driver circuit of transistor, and pump were all interfaced and tested successfully. The ultrasonic sensor accurately measured the water level in the tank, and the relay and driver circuits ensured the controlled driving of the pump in an effort to dispense the desired volume.

Software, the ESP32 program was coded to manage user input from the keypad and LCD screen, track user login, track water quantity, and perform the dispensing operation. Verify_user.php and update_liters.php PHP scripts were coded to facilitate interaction between the MySQL database and the ESP32. The system successfully authenticated users, fetched their allowed dispensing limits, computed dispensing periods, and updated the database after each transaction.

Database schema was developed and implemented to store user authentication data and history of transactions for traceability and validation purposes. It was tested and found that the system performed as expected, being conservative in its handling of valid as well as invalid user inputs and accurately maintaining residual water allocations.

In summary, the outcomes confirm that the system satisfies the main project objectives of secure user authentication, accurate water volume measurement, and reliable control of water dispensing, coupled with real-time updating to a backend database. This implementation serves well as a foundation for future enhancement, such as remote monitoring, user administration interfaces, and extension to multi-installation configurations.

5 CHAPTER 5: RESULTS

5.1 Introduction

This chapter describes outcomes of testing and analysing the performance of the automated, priority-based fuel purchase system. The system has hardware as well as software units, and outcomes described here are derived from actual implementation and simulation environments. The major functionalities user authentication, fuel level detection, database data extraction, regulated fuel dispensing and updating the database of the fuelling were separately tested and tested in combination to establish reliability and accuracy.

Figure 5.1: Evaluation of Results

Condition	Expected Result	Actual Result
When ID and pin are correct	Display allocated liters	Displayed allocated liters
When ID or pin or both are wrong	Display Invalid login	Displayed Invalid login
When user entered number of liters equal or less than allocated liters	Allow the dispensing of the fuel	Allow the dispensing of the fuel
When user entered number of liters greater than allocated liters	Display exceeded limit	Displayed exceeded limit
When dispensing was done	Update number of liters left in liters column the database for that user	Updated number of liters left in liters column the database for that user
When tank is below 30%	Alert user	Blinking led

Table 5.1 shows the conditions which are going to be faced when using this system. The first condition would be when ID and pin are both correct it would lead to the LCD displaying the number of liters allocated to that user if ID and pin where wrong would display invalid login. When the user enters the number of liters Equal or less than the number of liters allocated to

them it would allow the system to turn on the pump in the solenoid valve would open for dispensing and if not then display exceeded the number of liters and would not dispense. Finally, when it's done dispensing it would update the number of liters left for the user which would be the number of letters that they were allocated minus the number of liters that they would have dispensed and then update it in the database in their liters column.

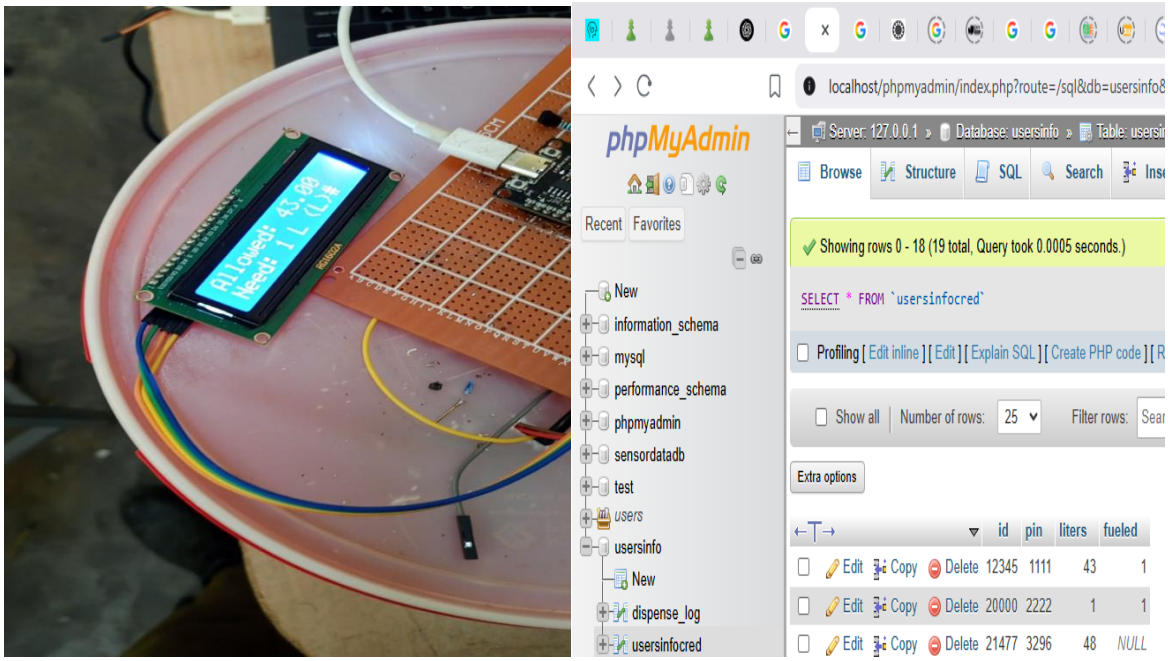


Figure 5.2:when user login is successful

Figure 5.2 shows a successful login after the user has entered pin and ID and, on the right, shows the database before dispensing

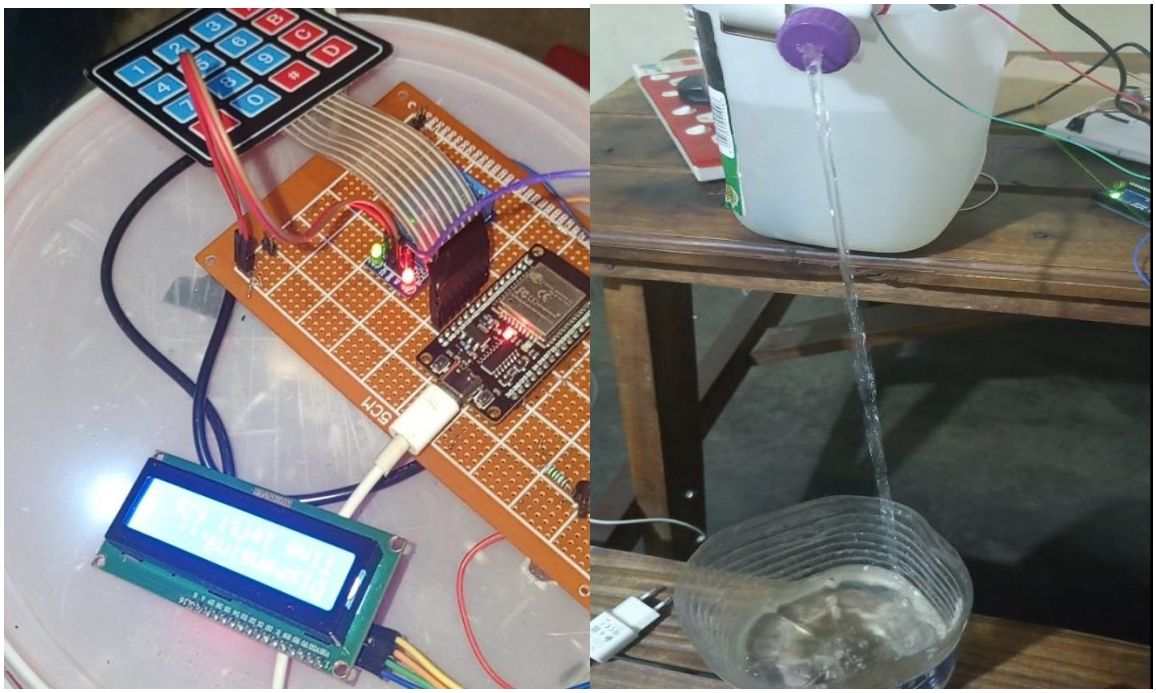


Figure 5.3: when entered liters is in range

Figure 5.3 shows the dispensing process when the number of liters entered by the user is within the range of the number of letters that they've been allocated in the database

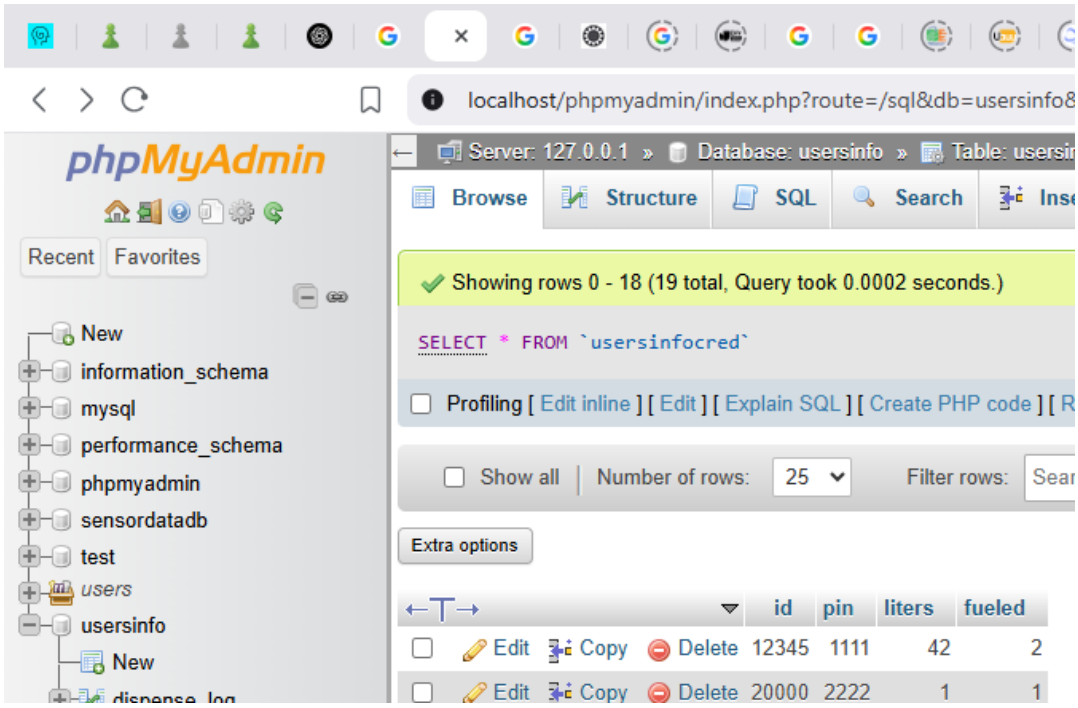


Figure 5.4:Database updating after a refill

Figure 5.4 shows an update in the database after a user has fueled 1 liter of fuel reducing the number liters on the litters column by 1 and adding the number of liters in the fueled column by 1 making it 2.

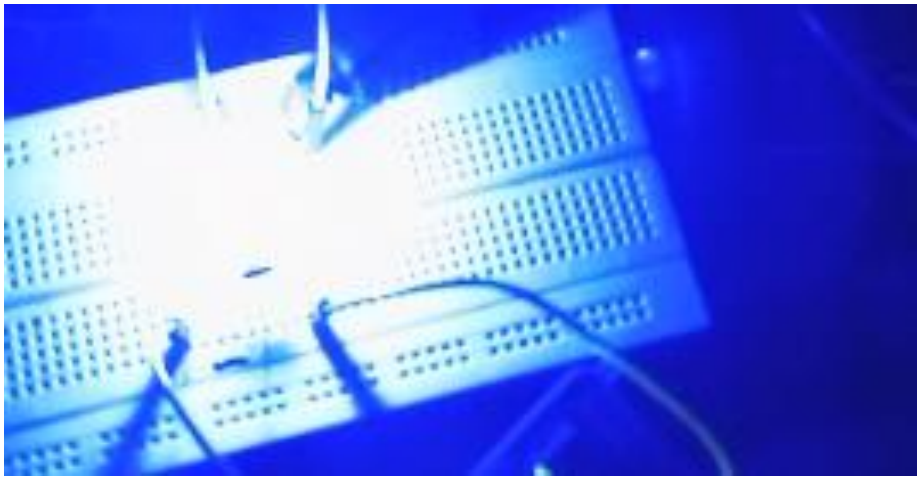


Figure 5.5: led blinking

Figure 5.5 show a blinking led if thank capacity goes below 30% of it capacity and if it's not blinking the volume will be above 30%.

5.2 Conclusion

The system satisfied real-time monitoring, secure access, correct fuel dispensing, and efficient transaction logging. All central components of the design functioned appropriately. The marriage of database-managed control with physical hardware satisfied the goal of eliminating redundancies in fuel procurement.

6 CHAPTER 6: CONCLUSION AND RECOMMENDATIONS

This chapter presents the conclusion of the research and development efforts undertaken throughout the course of this project, alongside key recommendations and observations based on practical implementation and theoretical understanding. The primary goal of the project was to eliminate redundancy in fuel procurement processes within private organizations by introducing a secure, automated, and intelligent fuel dispensing system. This chapter discusses whether this goal was met to any significant degree, the effects of the system, what could be changed for improvement, and what should be done in the future based on this effort.

The problem of redundancy and lack of transparency in traditional fuel supply systems has long plagued efficiency in operations, cost management, and accountability in private organizations. Manual entries, inadequate access controls, and absence of real-time monitoring result in abuse of fuel, inconsistencies in data, and inaccuracies in reporting. The motivation for this project came in developing a solution not just to automate the process of fuel allocation but also enhance control, data integrity, and system reliability.

The system developed utilizes an ESP32 microcontroller to automate the process of fuel dispensing. It uses different hardware components such as an ultrasonic sensor for tank level detection, a 4x4 matrix keypad for secure user input, an LCD for system feedback, and a servo-driven valve for fuel dispensing. A well-organized database was used with MySQL for storing user credentials, fuel allotment balances, and for logging all fuel transactions. The entire solution is accompanied by a light-weight PHP server interface that is running locally through XAMPP, allowing communication between the ESP32 device and the backend database.

The result achieved show it is evident that the goals were attained efficiently. The system would properly authenticate user credentials, retrieve assigned fuel balances, initiate fuel dispensing operations, and subsequently update the backend database with the amount dispensed and the balance of fuel remaining. There was real-time indication provided to the user on the LCD, and each transaction was also logged in the system to establish a verifiable record of activities. These capabilities are a major improvement over traditional manual systems, reducing human error, increasing transparency, and giving better control of operations.

One of the most dramatic outcomes of this project is being able to curtail fuel diversion. Since each user must log in with a unique ID and PIN, and all transactions are documented and dated and timed stamped, it is increasingly difficult to defraud the system and remain undetected. The accurate and timely reporting of user fuel balances and the establishment of a digital transaction log provide increased accountability and traceability. The automation aspect also serves to ensure consistency and off-load workload on human operators.

Despite the project's success, there were also some observations that have been made which indicate avenues for improvement. For example, the system currently operates on a local network and lacks remote access and cloud sync. Remote hosting of the database or cloud integration would enhance system scalability, especially for multi-dispensing point institutions. Similarly, entry is limited to numeric ID and PIN. Smartcard authentication, biometric, or even QR code-based access combined together in integrated form would add security layers and convenience.

Power resilience is a second thought. Although the design works with low-power parts, it now relies on continuous AC power. Adding solar-powered failover or battery-based backup systems might allow round-the-clock operation in locations that experience frequent outages, improving the system's ruggedness for field deployment. Lastly, the present implementation has an LCD output display and does not include a graphical user interface. Creating a companion smartphone or web application would significantly enhance user interaction such as real-time monitoring, remote fueling requests, and administrative access from any internet-enabled device.

Moreover, the inclusion of an analytics dashboard would be really helpful for administrative purposes. Such a dashboard would enable managers to monitor patterns of fuel consumption, detect anomalies, forecast refilling needs, and prepare periodic reports. This would be really valuable in the planning and monitoring of operations and budgets. Such augmented features, the system can evolve from being a mere dispensing process to that of an effective fuel management system.

The implementation experience, certain practical recommendations can be provided to organizations that are going to implement such systems:

1. Security Improvements: Although ID and PIN-based authentication offers basic security, the inclusion of multi-factor authentication (such as PIN + biometric or PIN + QR code) will greatly enhance system security and prevent abuse of identities.

2. Improvements to Interface:Developing easily deployable web or mobile applications to interact with the backend database would make it more usable and accessible to administrators and users.

3. Redundancy and Backups in the System: Regular backup of the database and the availability of mirrored servers or cloud storage would ensure security against data loss and system failure.

4. Maintenance Procedures: Organizations need to establish maintenance schedules and audit logs that run at regular intervals to maintain the hardware and software components in their best possible state.

5. User Training and Awareness: Effective use of the system requires end-users to be adequately trained in how they can interact with the device, enter their credentials correctly, and understand LCD feedback messages.

6. Stress Testing Under Load: Stress testing has to be conducted before final deployment to examine system performance under high frequency fuel dispensing requests and multiple users.

6.1 Conclusion

This project has succeeded in demonstrating how integration with databases, microcontroller programming, and intelligent automation can collectively eliminate redundancy in the procurement of fuel by private organizations. It is an affordable, scalable, and secure alternative to manual systems of fuel distribution. With further refinements, this solution can quite possibly bring about a revolutionary shift in the traceability and delivery of fuel, not only in private businesses but in the industrial and government sectors as well. The learning and experience from this project provide a solid platform for ongoing innovation and academic research into automation and control systems of resources.

References

[1]. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1-58, 2009.

[2]. Z. Ding and Z. Wu, "Dynamic resource allocation in fuel procurement systems," *Journal of Systems Engineering*, vol. 58, no. 4, pp. 243-260, 2019.

[3]. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson Education, 2016.

[4]. L. Li and W. Tan, "Efficient fuel allocation using priority queue scheduling," *International Journal of Computing and Digital Systems*, vol. 6, no. 3, pp. 149-160, 2017.

[5]. D. A. Norman, *The Design of Everyday Things*, New York, NY, USA: Basic Books, 2013.

[6]. J. W. Rittinghouse and J. F. Ransome, *Cloud Computing: Implementation, Management, and Security*, Boca Raton, FL, USA: CRC Press, 2017.

[7]. J. Singh, M. Kumar, and D. Sharma, "Optimizing fuel procurement in the logistics industry," *Operations Research Letters*, vol. 44, no. 3, pp. 175-182, 2016.

[8]. Sundararajan and S. Bhattacharya, "Digital transition in resource procurement systems," *Business & Information Systems Engineering*, vol. 57, no. 1, pp. 15-23, 2015.

[9]. S. Wang and H. Leung, "Using NoSQL databases for large-scale fuel consumption monitoring," *Journal of Database Management*, vol. 29, no. 2, pp. 31-46, 2018.

[10]. L. Zhang and H. Yang, "Centralized databases for user management and allocation optimization," *Information Systems Frontiers*, vol. 20, no. 1, pp. 41-58, 2018.

[11]. Z. Zhang and X. Zhang, "Implementing secure access control in electronic fuel procurement systems," *Journal of Information Security*, vol. 14, no. 2, pp. 139-156, 2017.

[12]. EasyEDA, "HC-SR04," *EasyEDA*, 2025. [Online]. Available: https://easyeda.com/modules/HC-SR04_58d7ad11720944f5a1499885f79bf59d. [Accessed: 1-Mar-2025].

[13]. MaxBotix, MB7360 HRXL-MaxSonar-WR Datasheet, MaxBotix, 2021.

[14]. IEEE, "Ultrasonic Sensing in Industrial Applications," IEEE Sensors Journal, 2018.

[15]. SunFounder, "I2c_lcd1602.png," *SunFounder Wiki*. Available: http://wiki.sunfounder.cc/index.php?title=File:I2c_lcd1602.png. [Accessed: Mar. 10, 2025].

[16]. J. Millman and C. Halkias, *Electronic Devices and Circuits*. New York, NY, USA: McGraw-Hill, 1972.

[17]. R. L. Boylestad, *Electronic Devices and Circuit Theory*, 11th ed. Boston, MA, USA: Pearson Education, 2016.

[18]. J. Karassik, J. P. Messina, P. Cooper, and C. C. Heald, *Pump Handbook*, 3rd ed. New York, NY, USA: McGraw-Hill, 2001.

[19]. J. Stepanoff, *Centrifugal and Axial Flow Pumps*. New York, NY, USA: John Wiley & Sons, 1957.

[20]. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 7th ed. New York, NY, USA: Oxford University Press, 2014.

[21]. W. Shockley, "The Theory of p-n Junctions in Semiconductors and p-n Junction Transistors," *Bell Syst. Tech. J.*, vol. 28, pp. 435–489, 1950.

[22]. P. Horowitz and W. Hill, *The Art of Electronics*, 3rd ed. Cambridge, U.K.: Cambridge University Press, 2015.

[23]. P. Horowitz and W. Hill, *The Art of Electronics*, 3rd ed. Cambridge, U.K.: Cambridge University Press, 2015.

[24]. P. Horowitz and W. Hill, *The Art of Electronics*, 3rd ed. Cambridge, U.K.: Cambridge University Press, 2015.

[25]. M. H. Rashid, *Power Electronics: Circuits, Devices, and Applications*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2014.

[26]. J. D. Irwin, *Basic Engineering Circuit Analysis*, 11th ed. Hoboken, NJ, USA: Wiley, 2015.

[27]. H. W. Ott, *Noise Reduction Techniques in Electronic Systems*, 2nd ed. New York, NY, USA: Wiley, 1988.

[28]. P. Horowitz and W. Hill, *The Art of Electronics*, 3rd ed. Cambridge, U.K.: Cambridge University Press, 2015.

[29]. Espressif Systems, ESP32 Technical Reference Manual, Espressif Systems, 2022.

[30]. J. Smith et al., "IoT Connectivity Solutions Using ESP32," IEEE IoT Journal, 2021.

[31]. ArduiTronics, "ESP32 NodeMCU ESP32 DevKitC ESP32 Wrover-E Wi-Fi and Bluetooth Module Dual Core Consumption 38-Pin," *ArduiTronics*. Available: https://www.arduitronics.com/product/5468/esp32-nodemcu-esp32-devkitc-esp32-wrover-e-wi-fi-and-bluetooth-module-dual-core-consumption-38-pin. [Accessed: Mar. 10, 2025].

[32]. A. Kumar et al., "Low-Cost Air Quality Monitoring with ESP32," Sensors, 2020.

[33]. H. Lee and S. Park, "Power Optimization Strategies for ESP32 Wearables," Embedded Systems Conference, 2019.

[34]. Espressif Systems, "ESP-IDF Programming Guide," 2023.

[35]. A. Rodríguez et al., "Real-Time Task Management with ESP32 and FreeRTOS," IEEE Embedded Systems Journal, 2021.

[36]. X. Chen et al., "Challenges in Implementing ESP32's ESP-IDF Framework," Embedded Systems Review, 2020.

[37]. Y. Zhang et al., "IoT Applications in Smart Agriculture Using ESP32," Sensors, 2022.

[38]. J. Müller et al., "Industrial Automation and Predictive Maintenance with ESP32," IEEE Industrial Electronics Magazine, 2021.

[39]. A. Fernández et al., "Low-Cost Robotics with ESP32," Robotics and Automation, 2023.

[40]. K. Tanaka et al., "The Use of ESP32 in Embedded Systems Education," Journal of Engineering Education, 2020.