ET161015_ET161034_MAK_Report_Updated

by liuc Central Library

Submission date: 13-Jan-2021 06:44AM (UTC+0300)
Submission ID: 1486678308
File name: ET161015_ET161034_MAK_Report_Updated.pdf (4.26M)
Word count: 16463
Character count: 81732

SMART DINING MANAGEMENT SYSTEM WITH BIOMETRIC AUTHENTICATION

by

SIBDAT BIN ALI

MOHAMMAD ABDUL WAZED

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering
INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

DECEMBER 2020

SMART DINING MANAGEMENT SYSTEM WITH BIOMETRIC AUTHENTICATION

by

SIBDAT BIN ALI

MOHAMMAD ABDUL WAZED

A project
submitted as partial fulfilment of the requirement for the degree of

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering
INTERNATIONAL ISLAMIC UNIVERSITY CHITTAGONG

DECEMBER 2020

CERTIFICATE OF APPROVAL

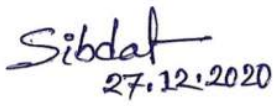
The project entitled as "**Smart Dining Management System With Biometric Authentication**" submitted by **Sibdat Bin Ali**, bearing Matric ID: **ET 161015** and **Mohammad Abdul Wazed**, bearing Matric ID: **ET 161034** of session **Autumn 2019**, to the Department of Electrical and Electronic Engineering, International Islamic University Chittagong, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Bachelor of Science in Engineering and approved for the examination held on **30th December,2020.**

12.01.2021

Supervisor
Mohammad Abdul Kader
Assistant Professor
Department of Electrical and Electronic Engineering
International Islamic University Chittagong.

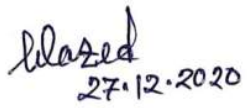
DECLARATION

It is hereby declared that this work has been done by us and no portion of the work contained in this thesis/project has been submitted elsewhere for the award of any degree or diploma.



_____

Sibdat Bin Ali



_____

Mohammad Abdul Wazed

ACKNOWLEDGMENT

In the name of Allah, the most gracious and the most merciful. First of all, we are grateful to almighty Allah for giving us strength and wisdom throughout all our life. We thank our family for their love, their moral and financial support they had given us. This helped us a lot. We would like to give our gratitude for our honourable supervisor Mohammad Abdul Kader, Assistant Professor, Department of EEE, IIUC who has given us significant suggestions and inspections during the whole process of the work. His invaluable help of constructive comments and suggestions throughout the experiment and project work have contributed to the success of this project.

Last but not the least we are really grateful to our teachers, friends, classmates, seniors and lab Assistants, who have given us their unlimited support and help in each and every aspect. We are really fortunate to have nice human being like them beside us.

Authors

ABSTRACT

In this era of technology, people are more involved in technology. The management of the hall dining system also needs to be modified with the advancement of technology. We have a project planned that will enable the hall dining system to be smart. Almost every hall dining system was followed by the traditional paper-based system. The main downside of this system is that papers can easily be lost or damaged. Also money and time are lost.

We have developed a fingerprint authentication system that will make it possible to be smart with the hall dining system. By design, once the device is switched on the system sets the time and date to be displayed. This fingerprint scheme allows the corresponding ID to be registered with the fingerprint and also provides the facility to match and delete the ID or fingerprint. To order the press button to register, match, delete and also enter an ID number, we used the keypad. When the fingerprint template is matched, the Servo motor allows the gate to be opened for the meal, and an SD card is used using the "Real Time Clock Module" to store the data with the corresponding time and date. A 16x2 Liquid Crystal Display is used to represent the user's behavior. Thus, our project will replace this manual method and make food distribution a more convenient and faster route. It will also help to decrease the fraudulency in the food distribution. All of this is done without investing a lot of money.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1: INTRODUCTION

People of all ages are more or less drawn to technology in this era of science, and our lives are becoming more dependent on science and people enjoy it. We employ technology in every area of our lives. A fingerprint-based food ordering system is integrated with the attendance system, enabling end users to register online to order food. By choosing the food items using the online web page, the user can see the food menu appearing online and order food products [1].

The conventional paper-based system is one of the commonly used hall dining systems. All the documents are kept on paper in this method. The primary downside to this method is that records can easily get lost or destroyed. Money, time and paper are also lost. No form of progressiveness is given by paper-based systems. Even a minor improvement involves reprinting the entire menu card. From the point of view of a consumer, this system is error-prone and time consuming since large employees are needed.

In this project, by making the system digital, we seek to ease the food distribution system of a dining room. Our main goal is to eradicate the primitive paper based token system for food distribution and replace it with Fingerprint authentication. This fingerprint system enables the corresponding ID to be registered with the fingerprint and also offers the facility to match and remove the ID or fingerprint. To register, match, delete and also enter the ID number, we used the Keypad to order the Press button. When the fingerprint template is matched, the Servo motor allows the gate to be opened for the meal. This procedure is saved in a SD card with corresponding time and date using Real Time Clock Module which is safe from unwanted disruption. A 16x2 Liquid Crystal Display is used to display the user's behavior. This can be used to display the day, hour, month, and year as a digital clock. It can therefore, act as both a security system and a clock. In addition to being light and lightweight, the system can be carried anywhere and can be quickly set up.

1.2: MOTIVATION

People are more involved in technology in the 21st century and our country is a developing country. So, our country should not lag behind technology. We need to design our technology with this target and bear in mind that technology should be readily available. We have planned a project that will enable the hall dining system to be smart. With the development of technology, the management of the hall dining system must also be updated. The conventional paper-based system was followed by almost every hall dining system the main drawback of which is that documents can easily be lost or damaged. Money, time and paper are also wasted. Our primary inspiration comes from our university's hall dining scheme. Students or employees have to wait a long time for a manual checking method, and after lunch, they often get late for their classes or jobs. Some also try to get food without giving cash or tokens.

1.3: OBJECTIVES

The project objectives are described below

- To remove the primitive paper based token system for food distribution.
- Using fingerprint verification to manage the food distribution of a dining.
- To prevent fraudulency in food management system.
- Fingerprints will be combined with student ID.
- A database is made for recoding and strong security measurement.

1.4: LEARNING OUTCOME

By doing this project we have learnt about

- Different types of sensors and their uses.
- Interfacing of sensors with Arduino UNO.
- Different types of functions and codes.

1.5: CHAPTER OUTLINE

During the planning and creation of this project six chapters has been covered. Chapter's contents are as follows:

- Chapter 1 (Introduction) The chapter presenting the description, inspiration and goal of the project.
- Chapter 2 (Literature Review) The chapter discussed previous work or research related to this project and comparisons with earlier work.
- Chapter 3 (Components) The components that were in this project have been discussed in depth in this chapter.
- Chapter 4 (System Design) The programming and interfaces between each module and Arduino are covered in this portion.
- Chapter 5 (Implementation and Result) is discussing about the implementation of the project and the results that we got from the project.
- Chapter 6 (Conclusion) Finally, this chapter addressed the description of this project in depth. The project's drawbacks and potential growth have been addressed.

CHAPTER 2
LITERARTURE REVIEW

2.1: INTRODUCTION

The conventional paper-based system is one of the commonly used hall dining systems. All the documents are kept on paper in this method. The primary downside to this method is that records can easily be lost or destroyed. Money, time and paper are also lost. No type of progressiveness is given by paper-based systems. Even a minor improvement involves reprinting the entire menu card. Since large manpower is needed, this system is prone to error and takes time from the point of view of a user [2]. Cooking and eating have been transformed by the creative use of new technologies, offering more opportunities. Our world is increasingly populated with a profusion of digital technologies designed to assist and automate more human tasks and activities. Furthermore, with embedded sensor-based and control systems, the physical world is becoming increasingly digitally instrumented and strewn. The kitchen and dining room, areas for family cooking and dining, have been a target setting for many ubiquitous machine prototypes exploring the use of technology in real-world conditions [3]. Previous research has concentrated on improving productivity and automation to bring technology into the domestic market.

Arduino may play an important role in food management and dining systems, in addition to various technical devices. The goal of Arduino is to make electronics accessible to all and to any consumer. Arduino is a freely accessible microcontroller program that has been popular with thinkers in the last decade. Embedded systems refer to devices that do not necessarily have a computational role in the engineering sense, but are managed by a computing entity. This processing unit may be a microprocessor, a Digital Signal Processor (DSP) or a microcontroller. Nowadays, such structures are so popular that in more ventures they are used than anyone would believe.

2.2: REVIEW OF PREVIOUS WORK

In this modern era research are going on to provide different updated ways to smart the dining management and daily new technology are coming to update our system. With this view various updated approaches are available in the food and dining management field and many more are in the process of study. Some of the previous works that were done based on food and dining management discussed below:

2.2.1: FOOD MANAGEMENT SYSTEM WITH FINGERPRINT AUTHENTICATION

In this method, the authors concentrate on a system that provides simple and safe features for managing the entire food order process through an online web portal.

The descriptions of the employees were kept here in the database along with their fingerprint templates. Whenever the finger is put, an individual unique code will be created for the employee every day. The admin will be provided with a special login feature so that the employee information can be accessed and updated and it also has the facility to adjust the e-menu or modify the e-menu, such as the name, price and food item details. It also has the capacity to demonstrate the food products ordered on the basis of employee ID. The system would also provide services dependent on the food items they purchased at the end of the month to produce their bills. This portal also provides the employee with a way to change their mobile number and also their e-mail ID on their own whenever they want, and they can also order food products on a time-based basis, it will be delivered after one hour of the deadline. The employee will also add the food items to the cart so that those food items can be processed or deleted and it also allows workers a way to enter their input on the quality of food and service, so that in the future orders we can fix those issues. The system would also help us to find out about the remaining food products that have yet to be shipped. The bills for the whole month will be measured and submitted to the e-mail ID of the employee and will be credited with the employee's monthly salary [1]. Diagram of the system block has been shown below in Fig 2.1.

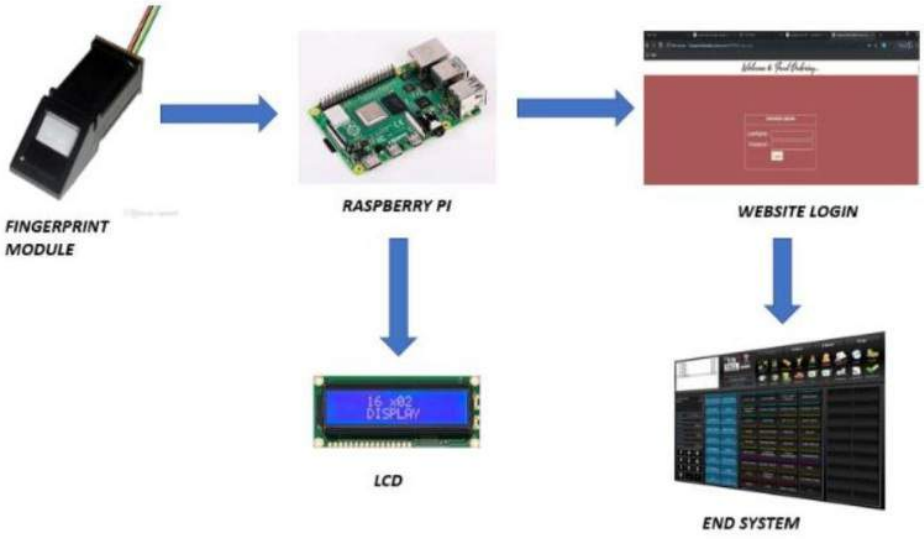


Fig 2.1: System block diagram [1].

In this block diagram, fingerprint module is used to scan the fingerprint, the person has to keep the finger in the module then the template is scanned and compared with the already stored template, these comparing of template is done in the raspberry pi kit. Once when the fingerprint template is get matched with the already existing template the massage "ID matched" is shown in the LCD. If it is not matched, then "Match not found" message is shown in the LCD. When the template is get matched the unique code will be generated and sent to the registered mobile number. Using the web portal, the e-food menu can be seen, and the required food items can be ordered, and the information's are get updated in the end system.

2.2.2: DIGITAL DINING RESTAURANTS USING ANDROID

In this system, the authors focus on a system that addresses the design and implementation of digital dining using Android technology in restaurants. This system is a simple dynamic utility system for databases that fetches all the data from a centralized database. The tablet at the client table contains all the restaurant and menu details of the Android application. The customer's tablet, kitchen display and cashier counter connect via Wi-Fi directly to each other. This wireless application is user-friendly, saves time, eliminates human mistakes, and offers customer reviews to improve quality and accuracy for restaurants. This system overcomes the shortcomings of previous automated food

ordering systems efficiently and is less costly as it requires a one-time investment in gadgets [2]. The system block diagram has been shown below in Fig 2.2.

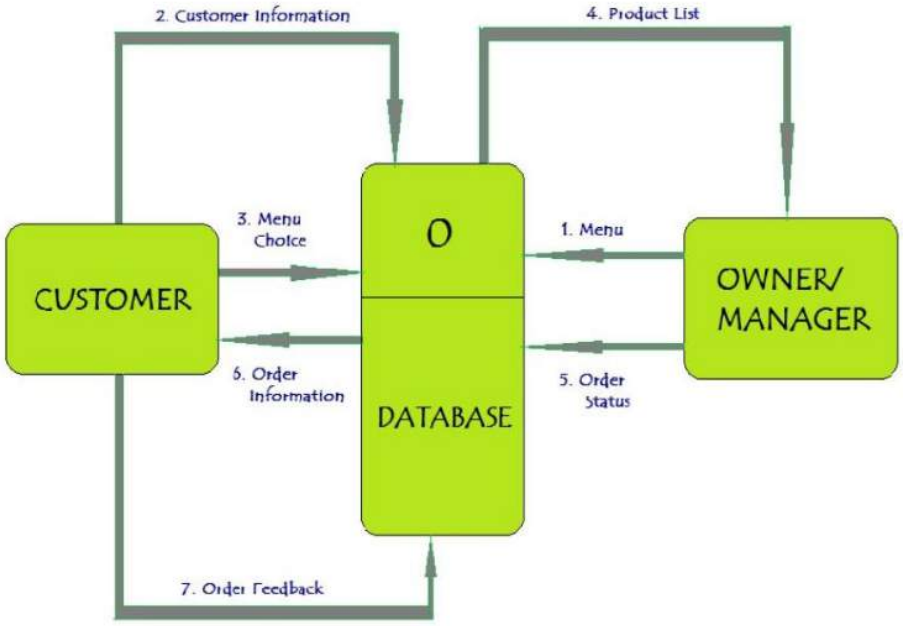


Fig 2.2: System block scheme [2].

First in this block diagram, the restaurant owner/manager will log into the system and change the menu according to the dishes' availability. The director will also market the different deals of the day. The details and menu choices selected by the client are transmitted over the wireless network to the system. The lists ordered from the system will be issued by the restaurant owner and the kitchen staff. The restaurant owner is able to check the system's order status. The customer will view the status of the order as well. The entire program will already be mounted on the tablets and held open on the tables.

2.2.3: CANTEEN MANAGEMENT SYSTEM USING RFID

This is a cloud, virtual cash, and RFID card based system. Cloud computing offers the benefits of auto-scaling, load balancing and greatly reduces the expense of hardware along with maintenance elimination. This software can be used on tablets, laptops and even smart phones. When extended to an important part of the workforce, this automation mechanism, i.e. "canteen," helps minimize service time, reduces queues, there is no pressure to provide the staff with the exact change for the order; to mention a few benefits

on the customer side of the canteen and on the other hand, it provides a secure way to store records and keep the money safe as most payments are made. In a day or month, the system will keep the total number of sales made by the canteen, which can be checked for analysis by the owner or the manager. The record of the purchased, used, remaining food and the amount spent on them is also kept. The benefit or loss that the corporation suffered is often shown. Since these information can only be entered by approved users, the chances of data misuse or data loss are almost reduced to zero [4].

2.2.4: CANTEEN AUTOMATION SYSTEM

This Canteen Automation System allows end users to register, read and pick food online from the e-menu card and order food online simply by choosing the food they want to use with the Android application. The results will appear directly on the screen near the chef, who will cook the food for you after choosing the food from the E-menu card. The framework is a mixture of both Android and Web services.

This method also significantly lightens the load at the end of the canteen, since the whole order taking process is automated. When an order is put on the webpage, it is entered into the database and then retrieved by a desktop program at the end of the canteen in nearly real-time. All products in the order are presented within this application, along with their corresponding choices and delivery information, in a succinct and easy to read manner. This enables canteen workers to go through the orders quickly when they are put and deliver the required items with minimal delay and uncertainty [5].

2.2.5: TOUCHPAD BASED FOOD ORDERING SYSTEM

An automated food ordering system is proposed in this paper that will smartly keep track of user orders. Basically, it's going to introduce a food ordering system for various types of restaurants in which users order or make custom food with just one click. The design of this framework will be carried out using the Tablet PC's Android program. The system architecture has been shown below in Fig 2.3.

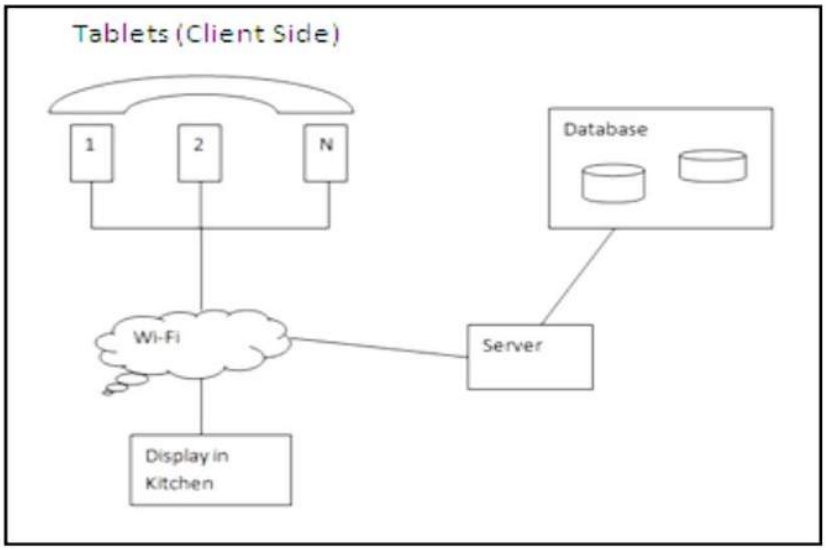


Fig 2.3: System architecture [6].

Customers order the food in this device by using the android-based touchpad. The system design, which includes three main areas of the restaurant, is shown in the figure: the serving area, the working desk of the restaurant owner (cashier table), and the kitchen. First the customer orders the food from the touchpad looking at different food combinations that are further brought to the kitchen to fulfill the order and the same is passed on to the tablet of each customer for billing [6].

2.3: COMPARISON WITH PREVIOUS WORK

In our project we have used fingerprint module which enables the corresponding ID to be registered with the fingerprint and also offers the facility to match and remove the ID or fingerprint. To register, match, delete and also enter the ID number, we used the Keypad to order the Press button. When the fingerprint template is matched, the Servo motor allows the gate to be opened for the meal. This procedure is saved in a SD card with corresponding time and date using Real Time Clock Module which is safe from unwanted disruption. A 16x2 Liquid Crystal Display is used to display the user's behavior.

- Low powered device.
- Complete automated system.
- User friendly.
- Future upgradable.

CHAPTER 3

COMPONENTS

3.1: INTRODUCTION

As our project aims at developing a smart food distribution system with biometric verification with a student database. To make this possible, an Arduino based circuit is created to manage the whole system. Here Arduino Uno is used to save an executing the program for controlling the whole system. As our system needs biometric security, we used R307 Fingerprint Module for registering and authenticating fingerprint of people, who would take meal by providing fingerprint. To input data for identifying fingerprint and security code we used 4×4 matrix keypad. When a fingerprint is authenticated on the system, it will show the ID of the fingerprint in a display so that we can determine if he is eligible or not. I2C LCD display is used for displaying data. A time log will also be saved on memory when fingerprint is entered on the system so that no misunderstanding happens. For this, we used DS1307 RTC Module. Finally after entering fingerprint if a person is eligible for taking food, then SG90 Servo Motor will power up and open a small gate in counter and food will be served through the gate.

3.2: LIST OF COMPONENTS

In this project several types of components were used. The components are listed below:

- Arduino UNO
- R307 Fingerprint Module
- 4×4 Matrix Keyboard
- DS1307 RTC Module
- I2C LCD Display
- SD card module
- SG90 Servo Motor

3.3: ARDUINO UNO

The Arduino Uno is an open source microcontroller board based on the microchip ATmega328P microcontroller developed by Arduino.cc. The board is equipped with input/output (I/O) digital and analog pin sets that can be connected to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (6 PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be operated by a USB cable or an external 9-volt battery, although voltages ranging from 7 to 20 volts are approved. For each I/O pin, the DC current is 40 mA and the 3.3V pin is 50 mA. The boot loader uses flash memory 32 KB and 0.5 KB memory, SRAM 2 KB, EEPROM 1 KB. The CLK's speed is 16 MHz. Digital pin 13 is an LED built in [7]. The figure of Arduino Uno has been shown below in Fig 3.1.

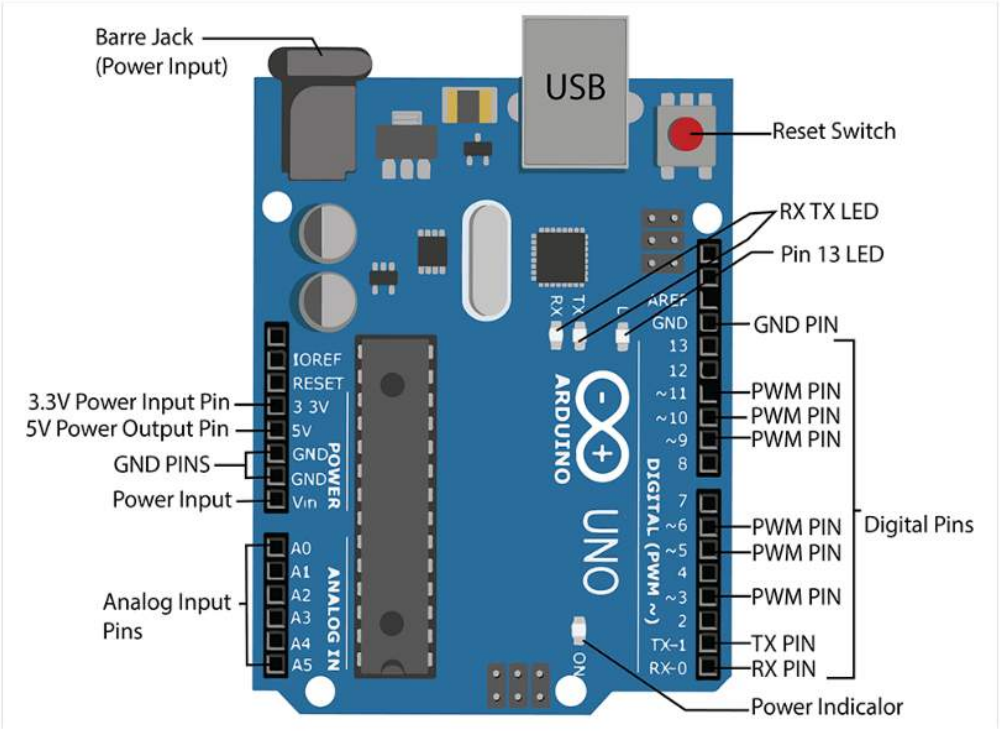


Fig 3.1: Arduino Uno [7].

3.3.1: PIN DESCRIPTION

The pin description of Arduino Uno has been shown below in table 3.1.

Table 3.1: Arduino UNO pin description.

Pin	Description
Digital Pins D0 to D13	Digital I/O Pin
Analog Pins A0 to A5	Analog I/O Pin
Digital Pin 3,5,6,9,11	Pulse Width Modulation (PWM) Pin
Pin#0 (Rx), Pin#1 (Tx)	Serial Communication Port
Pin#4 (SS), Pin#11 (MOSI), Pin#12 (MISO) and Pin#13 (SCK)	Communication Pin for SPI
Pin#A4 (SDA), Pin#A5 (SCA)	Communication Pin for I2C
Pin#13	Built in LED

3.3.2: COMMON COMMUNICATION PERIPHERALS OF ARDUINO UNO

The term involves the implementation on the board that is dedicated to a particular task that is unrelated to the CPU when talking about peripherals with respect to an Arduino or other microcontrollers. Peripherals can be thought of as units built into the Arduino boards that make specialized tasks simpler.

While several peripherals are included in the Arduino, we will look briefly at the widely used communication peripherals: UART, I2C, and SPI. These three serial data transmission formats are available on Arduino, although there are a range of I2C and SPI pins available for various Arduino models [8]. The common communication peripherals of Arduino UNO has been indicated below in Fig 3.2.

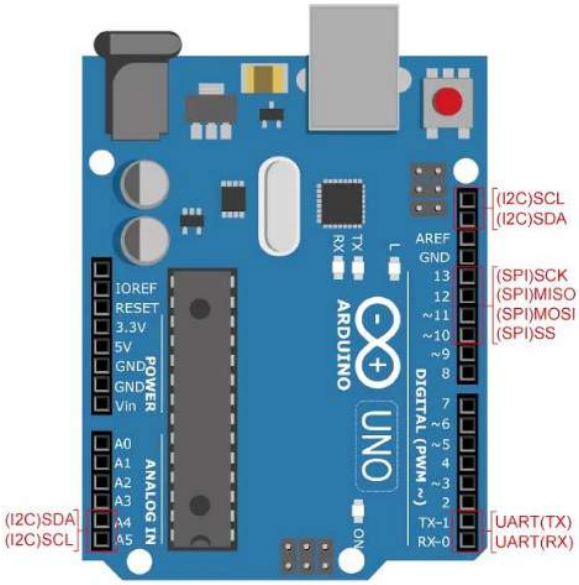


Fig 3.2: Common communication peripherals of Arduino Uno [8].

UART: It stands for Universal Asynchronous Reception and Transmission and is a fundamental protocol for communication that allows the Arduino to connect with serial devices. The UART device communicates via a USB port with digital pin 0 (RX), digital pin 1 (TX) and with various computers. On all Arduino boards, this peripheral helps the Arduino to directly communicate with a computer via its on-board USB-to-Serial converter. No clock signal is needed, unlike other protocols such as SPI and I2C, because the user gives the UART hardware the necessary timing information. The sampling process of UART communication has been shown below in Fig 3.3.

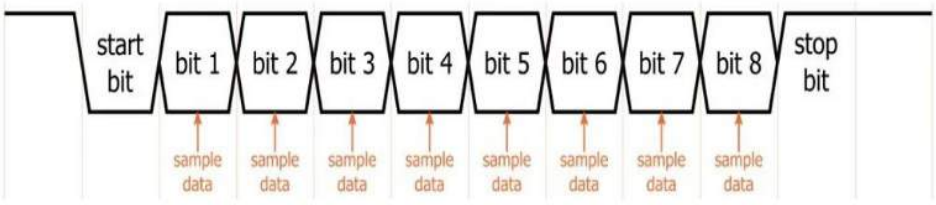


Fig 3.3: Sampling process of UART communication [9].

Start bit: The first bit of a one-byte transmission from UART. This shows that the information line is leaving its idle state. Usually, the idle state is logic high, so the start bit is logic low.

Stop bit: The last bit of a one-byte transmission from UART. Its logic level is the same as the idle state of the signal, i.e. logic high.

Baud rate: The estimated rate at which data can be transmitted (in bits per second, or bps). The frequency (in bps) corresponding to the time (in seconds) needed for transmitting one bit of digital data is a more accurate description.

In this case, synchronization and sampling can proceed as follows:

- The process of receipt is initiated by the dropping edge of the initial bit.
- To determine a sampling point that is near the middle of the bit duration, the receiver waits for 8 clock cycles.
- The receiver then waits for 16 clock cycles, taking it to the center of the first data-bit duration.
- In the receive register, the first data bit is sampled and processed, and then the module waits for 16 clock cycles before sampling the second data bit.
- This phase repeats until all data bits are sampled and stored, and then the UART interface is returned to its idle state by the rising edge of the stop bit [9].

I2C: This stands for inter-integrated-circuit, a protocol developed specifically for serial communication microcontrollers. While this peripheral is almost never used for communication between PC devices, modules and sensors are extremely popular, making it useful for projects that involve working together with several components. I2C allows you, potentially, to connect up to 128 devices to your main board.

When attaching two circuits to each other, think of the main computer as the 'master' and the attached components, such as sensors, pin extensions, and drivers as 'slaves'. I2C helps you to link multiple masters and slaves to your board while maintaining a clear communication route. A reliable communication path can be maintained because I2C uses an address system and a shared bus, which means that many devices can be connected to the same wires. The Arduino must, however, first select a specific device by transmitting a unique address before sending data. This gives any slave computer what

it needs while also assisting other masters. I2C uses fewer wires and all data is transmitted on a single wire, keeping the pin count low. The tradeoff for this simpler wiring is slower speeds than SPI.

SPI: SPI stands for Serial Peripheral Interface. Like I2C, SPI is a distinct form of serial communication protocol explicitly designed for microcontrollers to connect with each other. However, it has some primary differences from its I2C equivalent. The most obvious difference right off the bat is that with a maximum of four slave devices, SPI requires a single master device, while with I2C we can use several masters and slaves.

SPI is typically much quicker than I2C because of the simple protocol, and although data/clock lines are shared between devices, each device requires a unique address wire. SPI is typically found in locations where speed is important, such as with SD cards and display modules, or where information updates and changes quickly, such as with temperature sensors.

3.4: R307 FINGERPRINT MODULE

The R307 fingerprint module is a TTL UART interface fingerprint sensor that can be linked directly via the MAX232 / USB-Serial adapter to the UART microcontroller or to the PC. The user can store the fingerprint data in the module and can configure it in a 1:1 or 1:N mode to identify the person. The FP module can interact directly with a microcontroller that is 3.3 or 5V. A level converter is required to interface with a PC serial port. The R307 Fingerprint Module includes a high-speed DSP processor, high-performance fingerprint alignment algorithm, high-capacity FLASH chips and other hardware and software composition, secure execution, basic structure, fingerprint entry, image processing, fingerprint matching, stockpiling search and layout and different capacities [10]. The front and back view of R307 fingerprint module has been shown below in Fig 3.4.

Fig 3.4: R307 Fingerprint Module [10].

3.4.1: R307 FINGERPRINT MODULE FEATURES

- Power Supply : DC 4.2V-6V
- Current Consumption : ~50mA
- Interface : UART and USB
- Baud rate : (9600 * N) bps, N = 1-12, default is 6
- Image Acquiring Time : <0.5s
- Matching Modes : 1:1, 1:N
- Character File Size : 256 Bytes
- Template Size : 512 Bytes
- Storage Capacity : 1000
- Security Levels : 5
- FAR (False Acceptance Rate) : <0.001%
- FRR (False Recognition Rate) : <0.1%
- Average Searching Time : <1s (1:1000)
- Window Dimensions : 19 * 21 mm
- Working Environment : Temp = -10°C - +40°C, RH = 20%-80%
- Storage Environment : Temp = -40°C - +85°C, RH = <85%
- Outline Dimensions : Split Type, 44.1 * 20 * 23.5 mm

3.4.2: R307 FINGERPRINT MODULE PINOUT

The pinout diagram of R307 fingerprint module has been shown below in Fig 3.5 and pin description has been shown below at table 3.2.

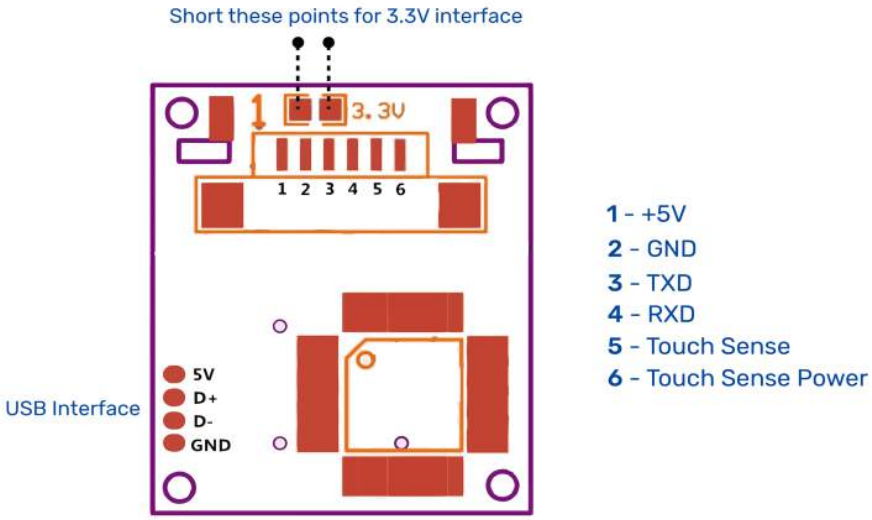


Fig 3.5: R307 fingerprint module pinout diagram [10].

Table 3.2: R307 fingerprint module pin description.

Pin Number	Name	Type	Description
1	+5V	IN	Positive Supply (DC 4.2V-6V)
2	GND	GND	Supply Ground
3	TXD	OUT	Data output (TTL)
4	RXD	IN	Data input (TTL)
5	Touch	OUT	Finger detection signal (max output current: 50mA)
6	3.3V	IN	Finger detection power (DC 3.3V-5V ~5uA)

3.4.3: WORKING PROCESS

The skin on our hands' palms has a special pattern called friction ridges that allow us to catch objects without efficiently slipping. These patterns consist of ridges and valleys arranged in certain arrangements for each person and are special. The working process of R307 fingerprint module has been shown below in Fig 3.6.

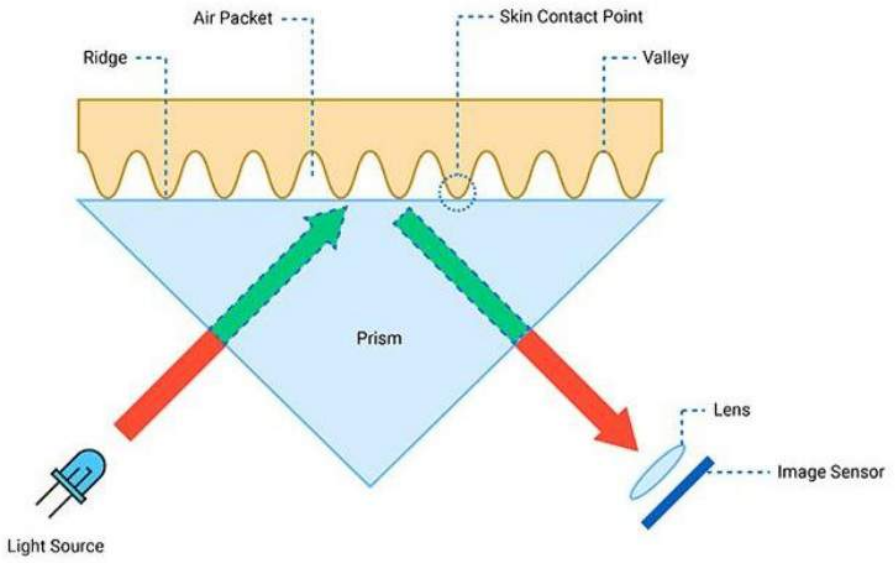


Fig 3.6: R307 fingerprint module working process [10].

An optical fingerprint scanner is based on the Total Internal Reflection concept (TIR). To allow TIR, a glass prism is used in an optical fingerprint scanner. Light from an LED (usually blue color) is allowed to enter through one face of the prism at a certain angle for the TIR to occur. The reflected light passes to the other face through the prism where a lens and an image sensor (essentially a camera) are located. Just the ridges make good contact with it as we hit a glass surface. The valleys remain separated by air packets from the earth. Our skin and air have various RIs and thus have a different effect on the evanescent area. This effect is called Internal Frustrated Absolute Reflection (FTIR). This influence affects the intensity of internally reflected light and is observed by the image sensor. The image sensor data is processed to produce a high contrast image that will be the digital representation of the fingerprint [10].

3.5: 4×4 MATRIX KEYPAD

The 4x4 matrix keypad is generally used in a project as an input. It has a total of 16 keys, indicating the same input values. A membrane switch is underneath any key. Each switch in a row is linked by a conductive trace beneath the pad to the other switches in the row. Each switch in a column is connected in the same way; a conductive trace connects one side of the switch to all the other switches in that column. For a total of 8 pins on a 4X4 keyboard, each row and column is moved to a single pin. The transition between a column and a row trace is closed by pressing a button, allowing current to flow between a column pin and a row pin [11]. The 4x4 matrix keypad has been shown below in Fig 3.7.

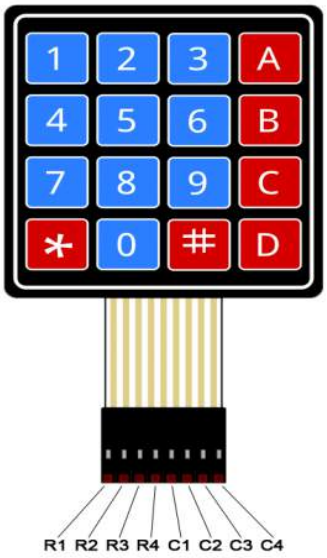


Fig 3.7: 4x4 matrix keypad [12].

3.5.1: 4X4 KEYPAD MODULE FEATURES

- Maximum Voltage across EACH SEGMENT or BUTTON: 24V
- Maximum Current through EACH SEGMENT or BUTTON: 30mA
- Maximum operating temperature: 0°C to + 50°C
- Ultra-thin design
- Adhesive backing
- Easy interface
- Long life.

3.5.2: 4X4 KEYPAD PIN CONFIGURATION

The pin description of 4x4 matrix keypad has been shown below in table 3.3.

Table 3.3: 4x4 matrix keypad pin description.

Pin Number	Description
ROWS	
1	PIN1 is taken out from 1st ROW
2	PIN2 is taken out from 2nd ROW
3	PIN3 is taken out from 3rd ROW
4	PIN4 is taken out from 4th ROW
COLUMNS	
5	PIN5 is taken out from 1st COLUMN
6	PIN5 is taken out from 2nd COLUMN
7	PIN5 is taken out from 3rd COLUMN
8	PIN5 is taken out from 4th COLUMN

The internal structure of 4x4 matrix keypad is shown below in Fig 3.8:

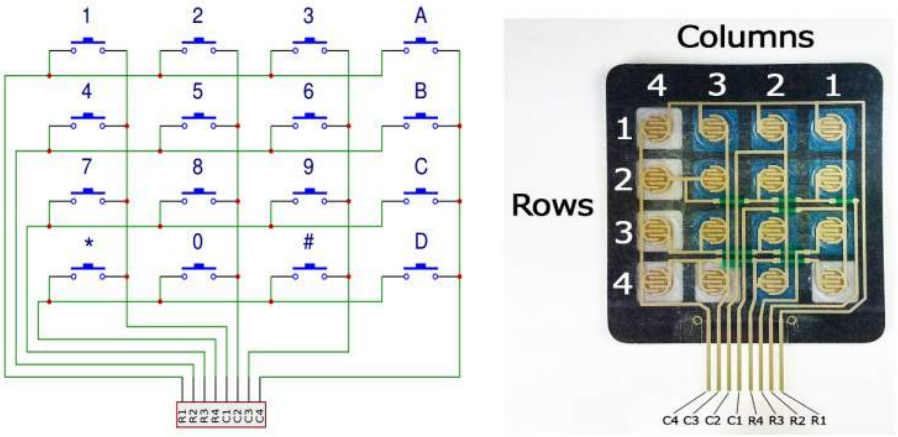


Fig 3.8: Internal structure of 4x4 matrix keypad [12].

3.6: DS1307 REAL TIME CLOCK MODULE

The DS1307 real-time clock (RTC) IC is an 8-pin unit that utilizes an I2C interface. The DS1307 serial real-time clock is a low-power, fully binary-coded decimal (BCD) clock/calendar plus 56 bytes of battery backup SRAM (RTC). The address and the data are transmitted serially via an I2C, bidirectional bus. The clock/calendar contains descriptions of the seconds, minutes, hours, day, week, month and year. The end date of the month will also be updated for months of fewer than 31 days plus changes in the leap year. With the AM / PM indicator, the clock is available in either a 24-hour or 12-hour setting [13]. The DS1307 has an integrated power sense circuit, which senses power failures and automatically turns into the backup device. The timekeeping process continues while the backup supply portion is running. Like a clock, they are available as ICs and track timing and are also available as a calendar. RTC's greatest advantage is its form of battery backup, which keeps the clock / calendar in operation even if the electricity fails. An extremely small current is required to keep the RTC animated. These RTCs can be used in many applications, such as motherboards and embedded systems, etc. DS1307 Real Time Clock (RTC) Module has been shown below in Fig 3.9.

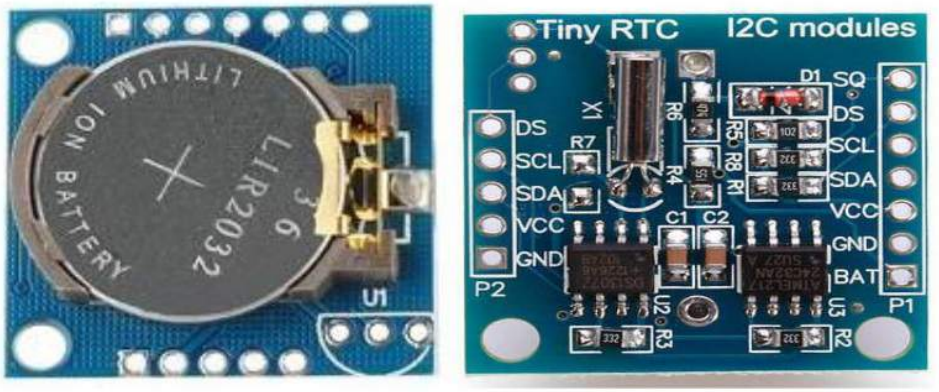


Fig 3.9: DS1307 Real Time Clock (RTC) Module [13].

3.6.1: FEATURES OF REAL TIME CLOCK

- Two-wire I2C interface.
- DS1307 based RTC with LIR2032 battery (Battery include).
- Available in 8-pin.
- 56-byte non-volatile RAM for data storage.

- The RTC counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation.
- 56 Bytes of Non-volatile memory available to the user.
- With the current time, the module is completely installed and preprogrammed
- Supply voltage is 5V DC.

3.6.2: PINS OF REAL TIME CLOCK

The pin diagram of RTC module has been shown below in Fig 3.10 and pin description has been shown below in table 3.4

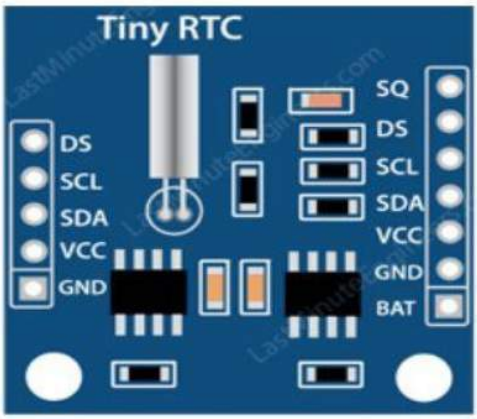


Fig 3.10: Pin Diagram of RTC Module [13].

Table 3.4: DS1307 RTC module pin description.

Pin	Description
X1 and X2	These are pins that require the internal oscillator to link the crystal of 32.768 KHz frequency. If X1 is attached to an external oscillator, X2 may be left floating
SDA	This is the I2C Interface Data Input / Output Pin.
SCL	Input Pin for Serial Clock. This is the I2C system clock reference panel.
SQW/OUT	Pin output of square wave. It can be left floating if it is not used.
VCC	Supply pin for the 5V DC.
GND	Pin for ground connection.
BAT	Supply Pin for Battery. For backup supply, we can attach to a 3V lithium battery.

3.7: I2C LCD DISPLAY

This is an I2C-interface 16x2 LCD display panel. You can display 16x2 characters on two sides, white on a blue backdrop. In general, Arduino LCD display projects would easily run out of pin assets, particularly with Arduino Uno. And with the wire soldering and attachment, it is also very complex. An I2C communication interface is used for this I2C 16x2 Arduino LCD Screen. It means that the LCD display requires only 4 pins: VCC, GND, SDA, SCL. On Arduino, it can save at least 4 digital/analog pins. All connectors are XH2.544 standard (Breadboard type). We can communicate directly with the jumper cable [14]. The I2C Liquid Crystal Display has been shown below in Fig 3.11.

Fig 3.11: I2C LCD Display [14].

3.7.1: I2C LCD DISPLAY FEATURES

- Compatible with Arduino/Genuino UNO, Leonardo, Mega, Micro, Nano, Mini
- I2C Address: It varies (Many use the default address of 0x27)
- Back lit (Blue with white char color)
- Supply voltage: 5V
- Interface:I2C/TWI x1,Gadgeteer interface x2
- Adjustable contrast
- Size: 80x36x20mmz(3.1x1.4x0.7in)

3.7.2: I2C LCD DISPLAY PINOUT DIAGRAM

The pinout diagram of I2C LCD has been shown below in Fig 3.12 and pin description has been shown below in table 3.5.

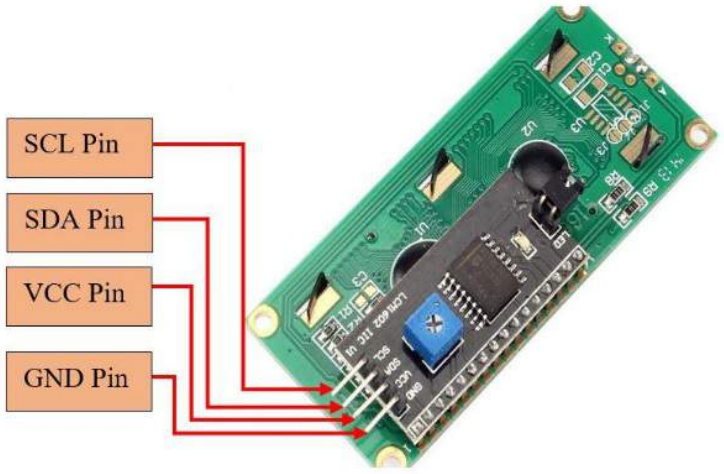


Fig 3.12: I2C LCD pinout Diagram [14].

Table 3.5: I2C LCD module pin description.

Pin	Description
SDA	This is the I2C Interface Data Input / Output Pin.
SCL	Input Pin for Serial Clock. This is the I2C system clock reference panel.
VCC	Supply pin for the 5V DC.
GND	Pin for ground connection.

3.8: SD CARD MODULE

There are two key components in the micro SD card module that certainly make it easy to add data logging.

- Any standard micro SD card's operating voltage is 3.3 V. Therefore, we cannot connect it directly to 5V logic circuits. Indeed, any voltage above 3.6V will permanently damage the micro SD card. This is why the module has an on-board ultra-low dropout controller that converts voltages from 3.3V to ~ 3.3V to 6V.
- There is also a 74LVC125A chip on the board that translates the interface's logic from 3.3V-5V to 3.3V. This is called Logic Level Shifting. That means we can interact using this surface with both 3.3V and 5V microcontrollers such as Arduino.

Currently, SPI mode and SDIO mode are two ways to interface with micro SD cards. Each SD card module is based on SPI mode, which is simple for any microcontroller to use, 'lower speed & less overhead' [15]. The SD card module has been shown below in Fig 3.13.

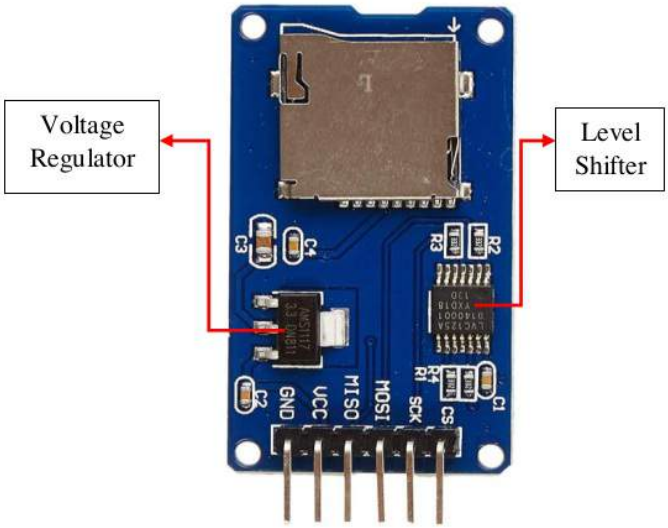


Fig 3.13: SD card module [15].

3.8.1: I2C LCD DISPLAY PINOUT DIAGRAM

The pin diagram of SD card module has been shown below in Fig 3.14 and pin description has been show below in table 3.6.

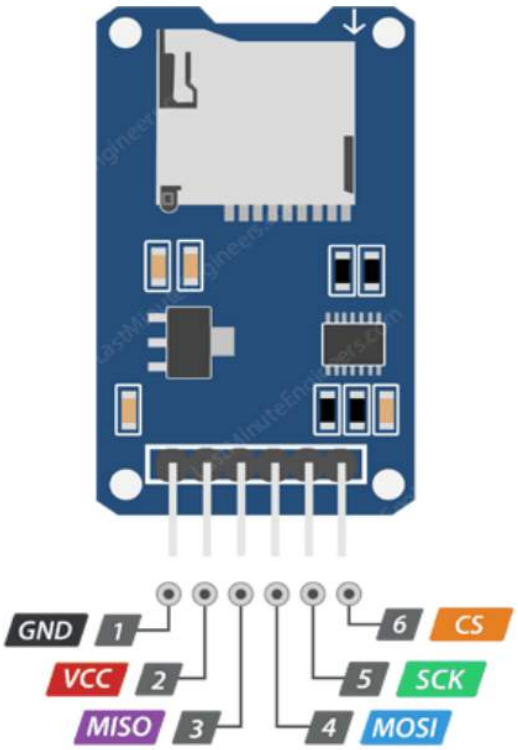


Fig 3.14: Pin diagram of SD card module [15].

Table 3.6: SD card module pin description.

Pin	Description
VCC	Supply pin for the 5V DC.
GND	Pin for ground connection.
MISO (Master In Slave Out)	It is the Micro SD Card Module's SPI output.
MOSI (Master Out Slave In)	The Micro SD Card Module is an SPI input.
SCK (Serial Clock)	This accepts clock pulses that synchronize the Arduino-generated data transmission.
SS (Slave Select)	Arduino (Master) uses it to allow and disable particular devices on the SPI bus.

3.9: SG90 SERVO MOTOR

A servo motor is a type of motor which can rotate very accurately. This type of motor usually consists of a control circuit that provides feedback on the motor shaft's current location, allowing the servo motors to rotate with great accuracy. If we want to rotate an object at a certain angle or distance, then a servo motor is used. It is simply composed of a basic motor that runs via a servo mechanism [16]. Usually, a servo motor comes with a gear arrangement that enables us in small and lightweight packages to get a very high torque servo motor. Because of these characteristics, they are used in many applications, such as toy cars, RC helicopters and aircraft, robots, etc. The servo motor SG90 has been shown below in Fig 3.15.

Fig 3.15: Servo motor SG90 [16].

3.9.1: SG90 SERVO MOTOR FEATURES

- Operating Voltage is +5V typically
- Torque: 2.5kg/cm
- Operating speed is 0.1s/60°
- Gear Type: Plastic
- Rotation : 0°-180°
- Weight of motor : 9gm
- Package includes gear horns and screws

3.9.2: SG90 SERVO MOTOR PINOUT DIAGRAM

The pin diagram of servo motor SG90 has been shown below in Fig 3.16 and pin description has been shown below in table 3.7.

Fig 3.16: Pin diagram of Servo motor SG90 [17].

Table 3.7: Servo motor SG90 pin description.

Wire No.	Wire Colour	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange/Yellow	PWM signal is given in through this wire to drive the motor

3.9.3: SG90 SERVO MOTOR WORKING PROCESS

The servo motor can control itself by sending a sequence of pulses to the signal line. A standard analog servo motor expects a pulse to be sent every 20 milliseconds or so (i.e. signal should be 50Hz) [17]. The pulse length specifies the position of the servo motor has been shown below in Fig 3.17

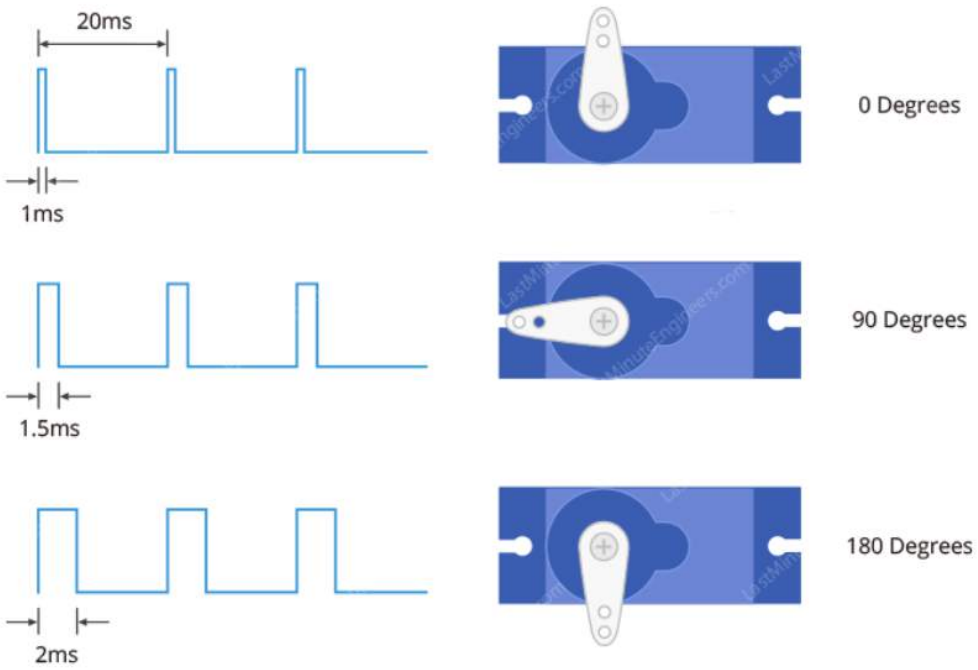


Fig 3.17: Working process of servo motor SG90 [17].

- If the pulse is 1 ms high, therefore the servo angle is zero.
- If the pulse is 1.5 ms high, the servo will be at its centre position.
- If the pulse is 2 ms high, the servo is 180 degrees.
- The servo shaft is driven over the entire 180 degrees of its movement by pulses varying from 1ms to 2ms.

CHAPTER 4

SYSTEM DESIGN

4.1: INTRODUCTION

This chapter contains a methodology to develop the prototype and described the system design of our project. Block diagram, hardware interfacing, circuit diagram, and flowchart of the system described below. This chapter shows how the Arduino UNO (microcontroller) connected with the sensors and how the data is collected from the system.

4.2: BLOCK DIAGRAM

The diagram depicts the components of a process with abstract, graphic symbols instead of realistic images. A model relies more on information interpretation and distribution than on functional operations. The diagram usually omits all information for this purpose that are not relevant to the data it attempts to convey, and can add unrealistic elements to support the understanding of the characteristics and relationships. The block diagram of the system has been shown below in Fig 4.1.

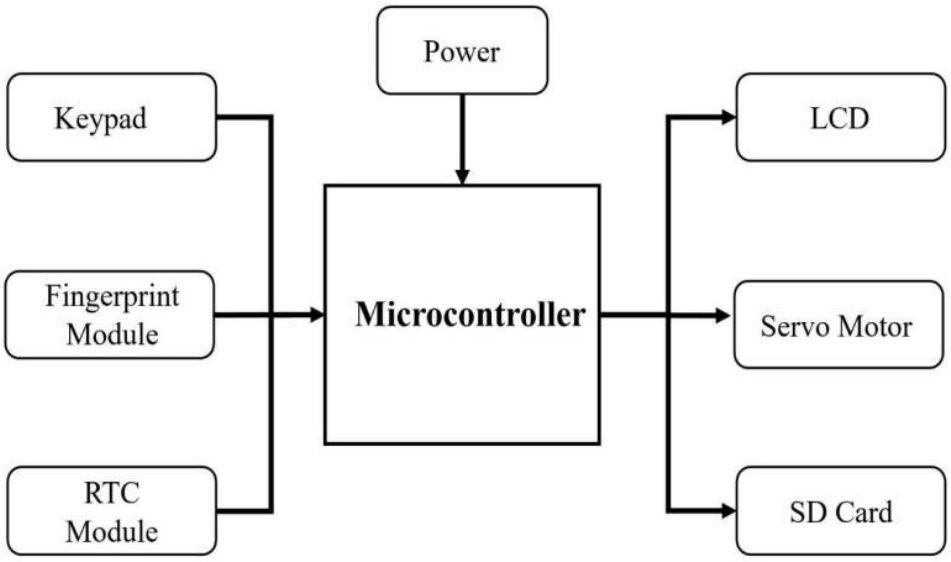


Fig 4.1: Block diagram of the system.

4.2.1: OPERATION OF BLOCK DIAGRAM

We have used an Arduino UNO as a microcontroller in this system. Keypad, Fingerprint module and RTC module has been used as input of the system and LCD, Servo motor and SD Card has been used as output of the system. 5V DC is supply to the microcontroller to run the system. By design, the system sets the time, day and date to be shown once the device is switched on. This fingerprint system requires the fingerprint to be registered with the corresponding ID and also provides the facility to match the ID or fingerprint and erase it. We used the Keypad to order the Press button to enroll, match, delete and also enter the ID number. The Servo motor allows the gate to be opened for the meal when the fingerprint template is matched, and an SD card is used to store the data with the corresponding time and date using the Real Time Clock Module. To demonstrate the user's behavior, a 16x2 Liquid Crystal Display is used.

4.3: DVELEOPMENT OF THE SYSTEM

In this section, we are going to discuss about the step by step development process of our system. Our system consists of microcontroller and several peripheral devices such as Fingerprint module, Keypad, SD card module, RTC module, Servo motor etc. The interfacing technique of different peripherals with microcontroller are discussed below:

4.3.1: INTERFACING FINGERPRINT MODULE WITH ARDUINO

We have used R307 fingerprint module which is a fingerprint sensor with a TTL UART interface for direct connections to microcontroller UART. We have used the Arduino UNO as microcontroller. In programming we have used the '*Adafruit*' fingerprint library function. The connection between Arduino and Fingerprint module is shown the below Fig 4.2

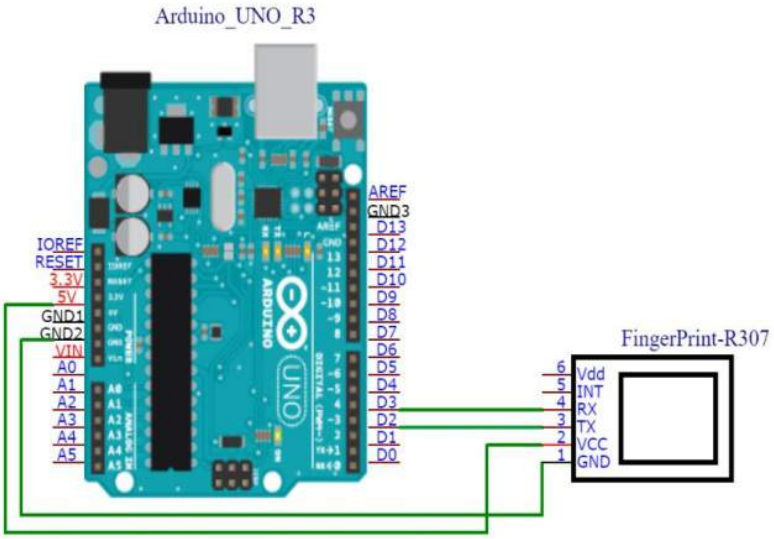


Fig 4.2: Connection between Arduino UNO and Fingerprint module.

Fingerprint sensor used several function for enrolling, matching and deleting the fingerprint such as;

- *SoftwareSerial mySerial(2,3);* Introduce the pin 2 as Rx pin which is connected to the fingerprint sensor's Tx pin and pin 3 as Tx pin of microcontroller which is connected to the fingerprint sensor's Rx pin.
- *Serial.begin(9600);* This means that the serial port can transfer data at a maximum of 9600 bits per second.
- *Finger.begin(57600);* means the UART is capable of transferring data a maximum of 57600 bits per second but the UART interface which supports baud rates up to 115200 bps.
- *readnumber(void);* this function use for entering number.
- *getFingerprintEnroll();* This function is use for enrolling and it's get the sensor parameter, it returns True if password is correct.
- *finger.getImage();* This means asking the sensor to take a picture of the finger pressed against the surface. And it's returning

FINGERPRINT OK on progress, FINGERPRINT NOFINGER if no finger is identified, FINGERPRINT PACKETRECIEVEERR on error in communication, FINGERPRINT IMAGEFAIL on error in imaging.

- *finger.image2Tz(1);* This means asking the sensor to convert the image to a template for the feature. And it's returning

  FINGERPRINT OK on success, FINGERPRINT IMAGEMESS if the picture is too messy, FINGERPRINT PACKETRECIEVEERR on error in communication, FINGERPRINT FEATUREFAIL on failure to recognize fingerprint features, FINGERPRINT INVALIDIMAGE on failure to recognize fingerprint features.

- *finger.image2Tz(2);* This means asking the sensor to convert the image to a prototype for the feature. And this time image has been converted.

- *finger.createModel();*It means asking the sensor to take two templates of the print feature and crate a model. And it's returning

  FINGERPRINT_OK on success , FINGERPRINT_PACKETRECIEVEERR on Communication error, FINGERPRINT_ENROLLMISMATCH on mismatch of fingerprints.

- *finger.storeModel(id);* This means asking the sensor for the measured model to be processed for later matching. And it's returning

  FINGERPRINT_OK on success , FINGERPRINT_PACKETRECIEVEERR on Communication error, FINGERPRINT_BADLOCATION if the location is invalid, FINGERPRINT_FLASHERR if the model couldn't be written to flash memory.

- *getFingerprintIDez( );* this function check the fingerprint is registered in scanner. And it returns

  FINGERPRINT_OK on success, FINGERPRINT_NOFINGER if no finger detected, FINGERPRINT_PACKETRECIEVEERR on Communication error, FINGERPRINT_IMAGEFAIL on imaging error.

- *getFingerprintID();* it's get the sensor parameter which is fingerprint ID, it returns True if password is correct.

- *finger.fingerFastSearch( );* It means asking the sensor to match saved templates by searching for the current slot fingerprint features. The matching position and the matching confidence in trust are stored in finger ID. And it's returning FINGERPRINT_OK on fingerprint match success, FINGERPRINT_NOTFOUND on no match made, FINGERPRINT_PACKETRECIEVEERR on Communication error.
- *deleteFingerprint(id);* this function is use for deleting fingerprint with corresponding FingerID.
- *finger.deleteModel(id);* means ask the sensor to delete the fingerprint. And it returns

  FINGERPRINT_OK on success , FINGERPRINT_PACKETRECIEVEERR on Communication error, FINGERPRINT_BADLOCATION if the location is invalid, FINGERPRINT_FLASHERR if the model couldn't be written to flash memory.

Here we also have used EEPROM library of microcontroller which allows us to save the data when the board is turned off like a tiny hard drive. This library contains some functions these are,

- *EEPROM.read(address);* Reads a byte from the EEPROM. Locations that have never been written to have the value of 255.
- *EEPROM.write(address, value);* Write a byte to the EEPROM. Here, address: the location to write to, starting from 0 (*int*). value: the value to write, from 0 to 255 (*byte*)
- *EEPROM.update(address, value);* Write a byte to the EEPROM. The value is written only if differs from the one already saved at the same address.
- *EEPROM.put(address, data);* Write any data type or object to the EEPROM.
- *EEPROM.get(address, data);* Read any data type or object from the EEPROM.

4.3.2: INTERFACING KEYPAD WITH ARDUINO

We have used the 4x4 Keypad in order to Press button to enroll, match, delete and also enter the ID number. The row pin of Keypad is connected to the digital pin 14,15,16,17 of microcontroller and column pin is connected to the digital pin 8,7,6,5 of microcontroller. The connection between Arduino and Keypad has been shown in below Fig 4.3

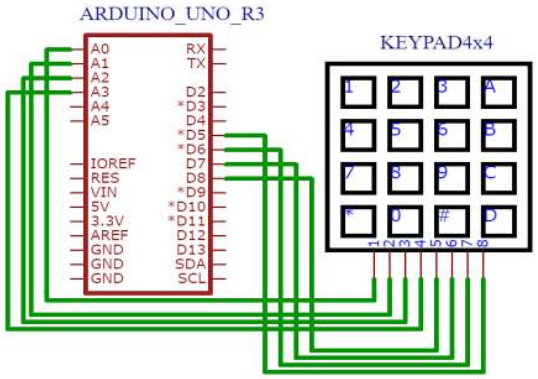


Fig 4.3: Connection between Arduino UNO and Keypad.

The design starts by including the library *'Keypad.h'* and defining the number of rows & columns on the keypad that we want to use.

We define a 2-dimensional keymap array *keymap[numRows][numCols]* which holds characters to be printed when a particular button is pressed on the keypad. Then we create an object of keypad library. The constructor *Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols)* takes five parameters.

- *makeKeymap(keymap)* initializes the internal keymap to be equal to the user defined keymap.
- *rowPins* and *colPins* are the arduino pins to which rows & columns of keypad connected.
- *numRows* and *numCols* are the number of rows & columns of the keypad.

Once a keypad object is created, we can issue a simple *getKey()* command to check which key, if any is pressed. There is another *waitForKey()* function that waits indefinitely for someone to press a key.

4.3.3: INTERFACING RTC MODULE WITH ARDUINO

To transfer data into microcontroller, the DS1307 RTC module uses the I2C communication protocol. The Arduino microcontroller always acts as the Master which is the responsible for clock signal and the DS1307 acts as the Slave in the Arduino Real Time Clock I2C interface. DS1307 RTC module has totally 12 pins form both the sides. Among these pins most important pins are SCL and SDA which are the reasons for I2C communication. Since we are using Arduino Uno it has separate pins for SCL and SDA. Therefore, the SCL and SDA pins are connected to the respective pins of Arduino which is shown in below Fig 4.4

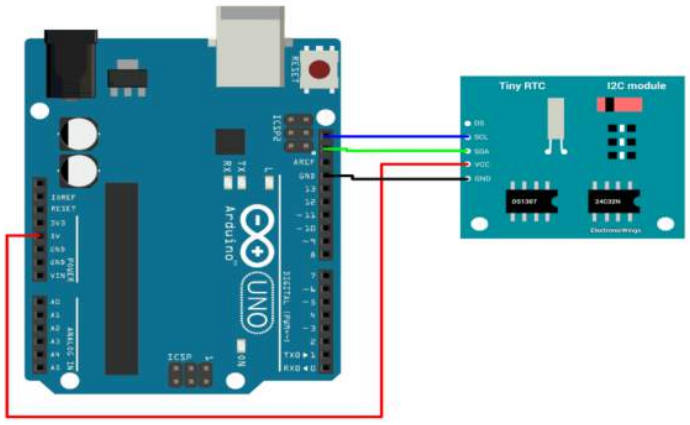


Fig 4.4: DS1307 RTC module interface with Arduino UNO.

To read the data and utilize, Arduino provides a special library called "*RTClib*" library and since the communication between Arduino and RTC module is an I2C communication a library called "*Wire.h*" also included. Which will allow to communicate with I2C devices. RTC is as slave address as 0x68.

After including the mentioned libraries then an object 'rtc' is created for RTC library and then the following functions were used to transfer necessary information to display:

rtc.begin(): It ensures that whether the RTC module is connected or not.

rtc.adjust(DateTime(F(__DATE__),F(__TIME__))): Which sets the current date and time in the respective sketch file.

now(): This function returns the current date and time and which will be stored in the variable called "DateTime".

now.hour(), now.minute(), now.second(), now.dayOfTheWeek(), now.day(), now.month(), now.year(): These functions return current hour, minute, second, day of the week, date, month and year respectively.

Then for proper I2C communication some functions belong to *Wire.h* library also used. The functions are as follow:

Wire.beginTransmission(RTC): This function begins the I2C transmission between Master and Slave devices with the given address RTC.

Wire.write(): This function takes the receives data from Slave device and writes the data according to the request from the Master device.

Wire.endTransmission(): The transmission which was begun by the function *beginTrasnmission()* will be ended by this function.

Wire.requestFrom(): This function requests Slave device for data in bytes. This function is accessed by the master device.

Wire.read(): This function read the bytes sent by the Slave device after the request from the master device.

Now the necessary data such as date, time, etc. are retrieved in the form of BCD it is mandatory to convert it to decimal for the purpose of display in the dot matrix display. Therefore, we have defined a separate function to convert from BCD to Decimal.

4.3.4: INTERFACING I2C LCD WITH ARDUINO

We have used the 16x2 LCD using I2C protocol. Wiring an I2C LCD is a lot easier than connecting a standard LCD. It needs to connect 4 pins instead of 12. Here also Arduino play the master and LCD is the slave. As Arduino Uno allows up to 128 slaves in I2C protocol using specific address. It use the library named as 'LiquidCrystal_I2C.h'. This library has many built-in functions that make programming the LCD quite easy. As we are using Arduino UNO it has separate pins for SCL and SDA. Therefore, the SCL and SDA pins are connected to the respective pins of Arduino which is shown below in Fig 4.5

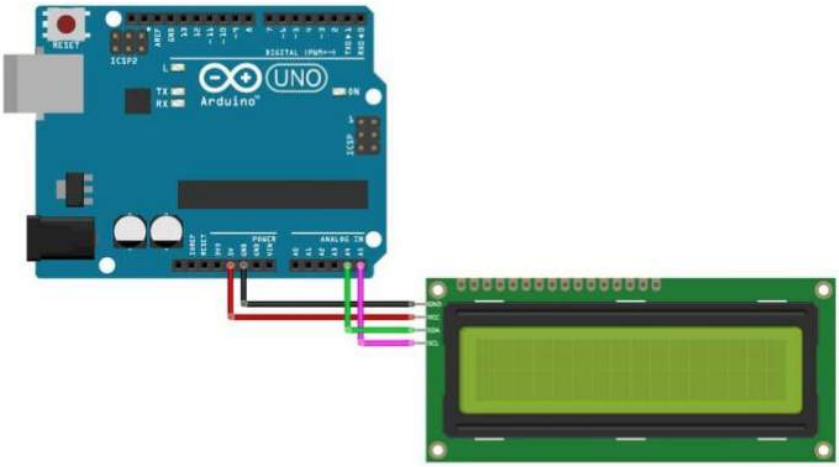


Fig 4.5: Interfacing I2C LCD with Arduino UNO.

With the LiquidCrystal_I2C class, the next step is to construct an LCD object and define the address and dimensions. We use the LiquidCrystal_I2C(address, columns, rows) function for this. When we use a 16-2 LCD, the line is changed to LiquidCrystal_I2C (0x3F,16,2). Here, the slave address is 0x3F. There are some default function we have used. Such as;

- *lcd*.clear();Clears the LCD screen and positions the cursor in the upper-left corner.
- *lcd*.home(); The cursor is placed on the upper left side of the LCD. That is, when outputting subsequent text to the display, use that spot. Use the *clear()* feature instead to clear the display too.
- *lcd*.cursor(); Display the LCD cursor: an underscore (line) at the position to which the next character will be written.
- *lcd*.setCursor(col, row); Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.
- *lcd*.write(data); Write a character to the LCD.
- *lcd*.print(data); and *lcd*.print(data, BASE); data: the data to print (char, byte, int, long, or string). BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Then for proper I2C communication some functions belong to *Wire.h* library also used that already discuss at section 4.3.3

4.3.5: INTERFACING SD CARD MODULE WITH ARDUINO

Each SD card module is based on the easy-to-use 'lower speed and less overhead' SPI mode for any microcontroller. The micro SD card module is reasonably easy to attach. It has six pins: Vcc, GND, MISO(Master In Slave Out) is SPI output from the Micro SD Card Module, MOSI(Master Out Slave In) is SPI input to the Micro SD Card Module, SCK(Serial Clock) pin accepts clock pulses synchronizing Arduino generated data transmission and Arduino(Master) is using SS(Slave Select) or CS(Chip Select) pin to allow and disable specific devices on SPI bus pins [15].

Before inserting the micro SD card into the module, we have properly formatted the card at FAT16. As micro SD cards need a lot of data transfer, when connected to the hardware SPI pins on a microcontroller, it will offer the best performance. The hardware SPI pins using another set of pins are much quicker than 'bit-banging' the device code. Digital 13 (SCK), 12 (MISO), 11 (MOSI) and 4 (SS/CS) are Arduino UNO boards with SPI pins. The connection of SD card module with Arduino UNO is given below in Fig 4.6

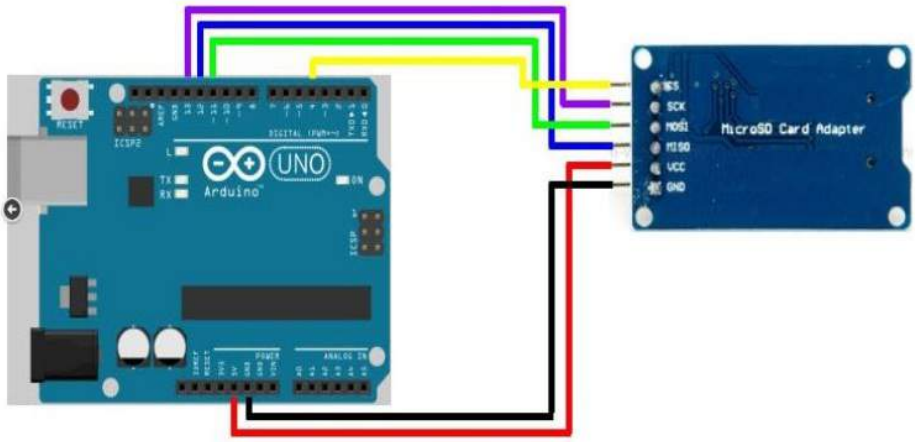


Fig 4.6: Interfacing SD card module with Arduino UNO.

The sketch begins with the built-in SD library (*SD.h*) and SPI library (*SPI.h*) in the case of programming, which helps us to quickly interact with the SD card over the SPI interface.

The next thing we do after the libraries have been included is announce the Arduino pin to which the SD card module's *chipSelect (CS)* pin is connected. Like all of the Arduino

digital pins, the CS pin is the only one that is not really set. Since we are using a hardware SPI interface, we do not need to declare other SPI pins, because these pins are already declared in the SPI library. After the pin is declared, we create the *myFile* object, which will later be used to store the data on the SD card.

Next, in the section *setup():* To display the results on the serial monitor, we start serial communication. Now, we'll initialize the SD card by using the *SD.begin()* feature.

Next, the function *SD.open()* will open a file called 'DATA.txt.' In our case, it will be generated as such a file is not present. The other FILE_WRITE parameter opens the file in the read-write mode.

If the file is opened, we write the data into the file using the *myFile.println()* function. After that, to ensure that the data written to the file is saved, we need to use the *close()* function.

4.3.6: INTERFACING SERVO MOTOR WITH ARDUINO

In our project, we used the SG90 servo motor, which allows the gate to open by moving its arm. By sending a series of pulses to the signal line, we power the servo motor. There are three colors of wire in the Servo motor. Between servo motors, the color of the wires varies, but the red wire is always 5V and either black or brown will be GND. Usually, the control wire is orange or yellow. The connection between servo motor and Arduino UNO is shown in below Fig 4.7

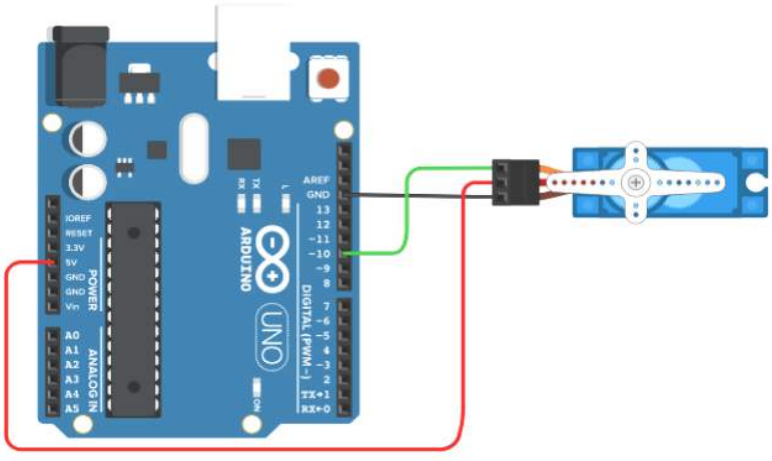


Fig 4.7: Interfacing Servo motor with Arduino UNO.

In case of programming, we have used servo motor library (*servo2.h)* to interface with it. There are some function we have used such as;

1. *Servo myservo;* This creates an object named myservo of the class Servo.
2. *myservo.attach(pin);*
    - This function attaches the servo variable to a pin.
    - Pin is the pin number to which the servo is attached.
3. *myservo.write(angle);*
    - This function writes a value to the servo, thus controlling the position of the shaft.
    - Angle can take values between 0 to 180.

4.4: CIRCUIT DIAGRAM

The figure of the circuit diagram are shown in below Fig 4.8 where it emphases the complete connection and describe which pin connect with which one and also shown, the component have how many pin.

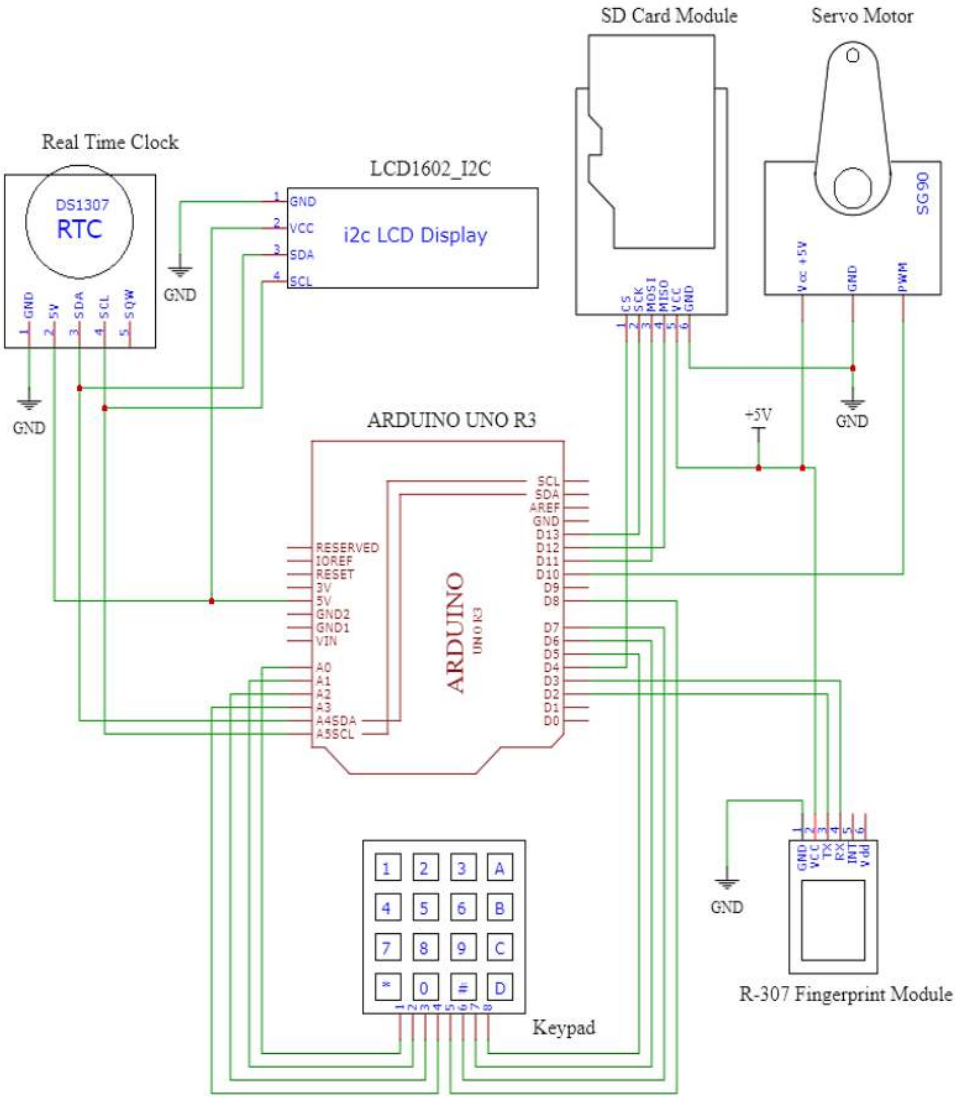


Fig 4.8: Circuit diagram of the system.

The Arduino Uno is an open-source microcontroller board based on the microchip ATmega328P microcontroller. The board is fitted with digital and analog input/output (I/O) pin sets that can be attached to different expansion boards (shields) and other circuits. In our project, we have used pin D2 as Tx and D3 as Rx for communicating with fingerprint module. The row pin of keypad is connected to the pin A0 to A3 and the column pin of keypad is connected to the pin D8 to D5. The RTC module and LCD are connected to SDA (Serial Data) and SCL (Serial Clock) pin of Arduino which are A4 and A5. The CS (Chip Select), SCK (Serial Clock), MOSI (Master Out Slave In) and MISO (Master In Slave Out) pin of SD card module are connected with pin D4, D13, D11 and D12 respectively. The control pin of Servo Motor is connected to the pin D10 of Arduino. Microcontroller and all peripheral device need 5V DC to run the system. Thus we have supplied 5V DC to circuit through USB cable using adapter.

4.4.1: CIRCUIT OPERATION

In our project, we have supplied 5V DC to circuit through USB cable using adapter. When microcontroller program is activating then circuit also activating. When circuit have command the area of microcontroller reader. Then the microcontroller sends the information to the microcontroller circuit in the main circuit and the main circuit transfers the information. We used the communication protocol UART (Universal Asynchronous Reception and Transmission) to read fingerprint module info. It is important to position a finger on the surface of the fingerprint module whenever the user registers, matches or removes his fingerprint. We have EEPROM (Electrically Erasable Programmable Read-Only Memory) of Arduino uno to store the fingerprint template. EEPROM is a non-volatile memory that protects the fingerprint template from microcontroller deletion by resetting or switching on/off. The I2C (Inter-Integrated-Circuit) microcontroller protocol is used by the RTC (Real Time Clock) module and the LCD. Therefore, both the RTC module and the LCD used separate address buses to communicate with microcontroller. The Arduino uno SPI (Serial Peripheral Interface) protocol was used on the SD card module. We used the Arduino uno PWM (Pulse Width Modulation) pin to operate the servo motor. To operate the system, the keypad is used to press the corresponding key.

We have programmed the system as whenever it switched on, to show time and date at LCD. In order to save the data, the system initializes the SD card module first. Then it

shows the keypad button 'B' to take the meal. If the user is registered with the respective ID, by opening the gate, it will allow the meal to be taken, but there is a requirement that the user takes only one meal at a time. Actually we built it in that way, by pressing the button 'C' of the keypad that opens a window to enter a particular administrative password, no one would be allowed by the system to take meal second time until the meal array cleared by administrator in the case of both lunch and dinner. There is an opportunity to register if the user is not registered, by pressing the 'A' button on the keyboard that opens a window to enter a particular administrative password. If the password is right, then there is an option to enter the corresponding ID that will be up to 127 with three digits. After that, some fingerprint registration instructions will be shown. There is an opportunity to delete the fingerprint from the computer by pressing the button 'D' of the keypad that opens a window to enter a particular password if anyone refuses to take meal or close meal account or administrator want to close someone's meal account. If the password is correct, allow the ID to be entered and deleted. In three separate phases of defense, we have three distinct passwords.

4.5: PROGRAMMING

We have used Arduino UNO use as our microcontroller. Arduino code or program is written in C++ with an addition of special methods and functions. C++ is a human-readable programming language. When we create an Arduino code file, it is processed and complied with machine language.

There are two main functions that execute the program. These are

- *void setup():* The void keyword is used only in function declaration. It returns no information to the function from it was called. The setup function initializes variables, pin modes, start using library etc. The setup function will only run once, after each power up.
- *void loop():* This function is used for running main program over and over again. This helps continuous execution of certain program simultaneously.

In our project, we have used various peripheral devices. To execute the system it needs different libraries and functions that we have discussed above at section 4.3 (Development of the system). The code of the system which has been loaded into microcontroller is given in the Appendix section.

4.5.1: FLOWCHART

The flowchart is represent the process of project. The program's flow chart is loaded into microcontroller has been shown in Fig 4.9, 4.10 and 4.11 below:

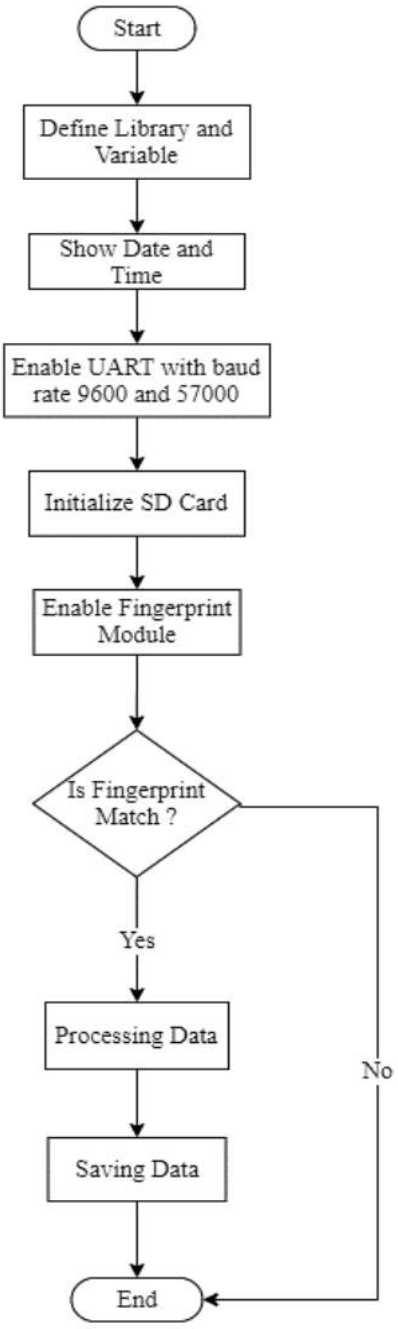


Fig 4.9: Flowchart of the system.

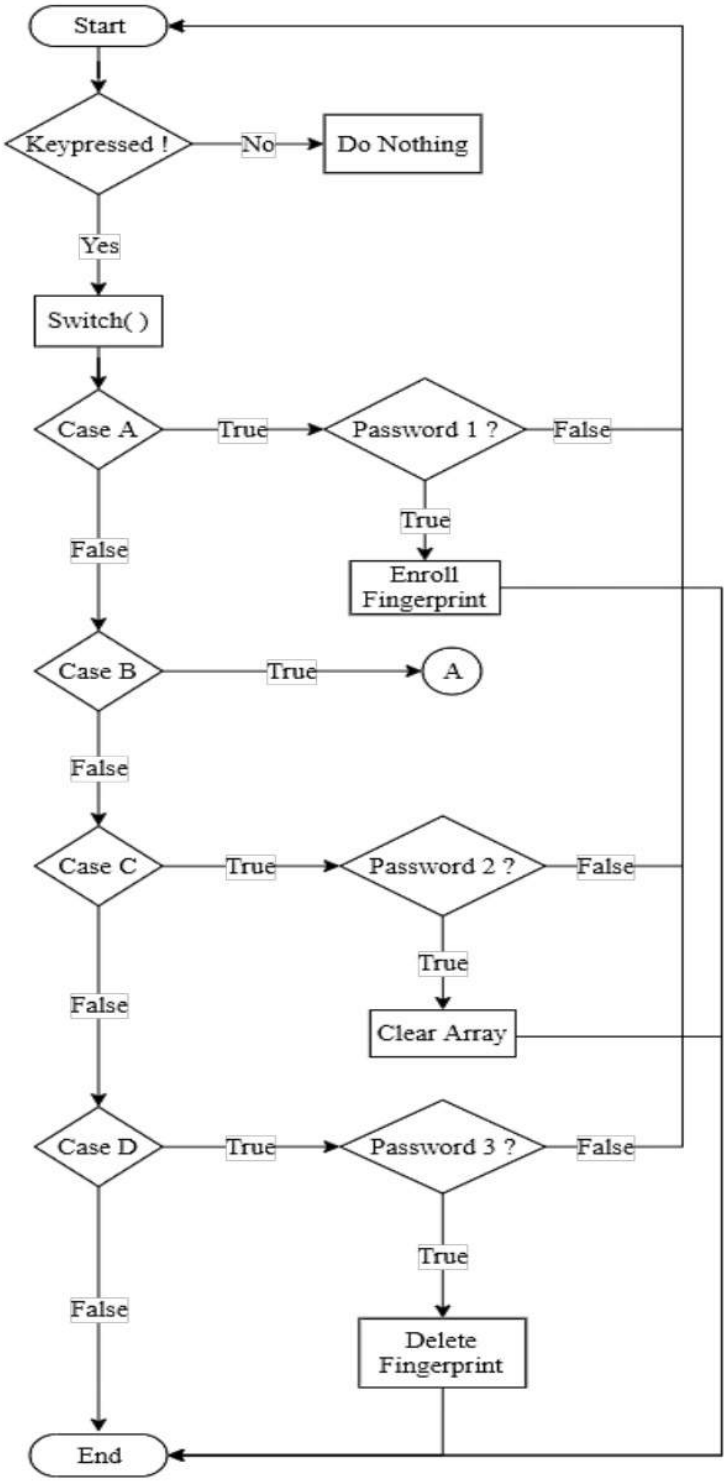


Fig 4.10: Flowchart of the response of corresponding Key.

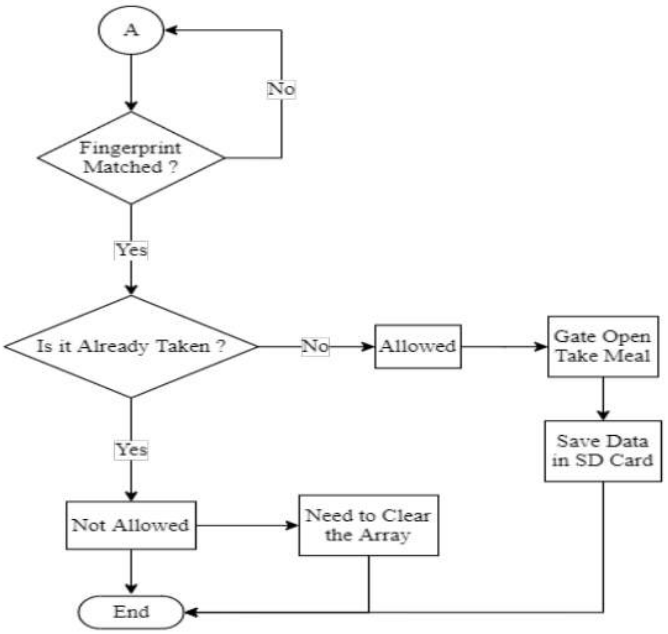


Fig 4.11: Flowchart of the response of matching Fingerprint.

In above Fig 4.9, we are showing the main flowchart of our system. First of all, we have defined the library and variable. The UART communication protocol is then enabled. After that we have initialized the SD card. We also have the fingerprint module activated to make the device ready to fit the fingerprint and use it to react. After that it processes the data and save the data.

The Fig 4.10 is about the flowchart for the response of pressing key on keypad. There are 4 alphabet keys in the keypad module: A, B, C and D. We have been using these keys for various purposes. Like registering fingerprint users, they have to press 'A' key, press 'B' key to take meal users, press 'C' key to clear the array from administrator and press 'D' key to remove any fingerprint.

The Fig 4.11 is about the flowchart for the response of matching the fingerprint of user. If it matches, then the device checking the ID from the Array either the user take the meal earlier or not. If the meal had already been taken, so it would not be permitted. If the user does not take meal earlier, then the gate to take meal can be opened and the gate will be automatically closed by itself after a few minutes. The system stores the user ID on the SD card with the corresponding date and time, and also displays the user's remaining meal.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1: INTRODUCTION

The system is designed to remove the primitive paper based token system for food distribution of a hall dining room. This is a fingerprint authentication system which maintain the security and food management. This system can store the data in corresponding action. LCD shows the action of user and also show the time and date when the device switched on. Keypad is used to Press the respective button for register, match, delete. The Servo motor allows the gate to be opened for the meal when the fingerprint template is matched.

5.2: IMPLEMENTATION

In the system all the sensors, LCD, Keypad and Arduino UNO are placed on a plastic box in the respective manner as prototype. The microcontroller was connected with the components through jumper wires. Since the sensors placed in such way, it can be easily either detached or mounted in case if they are to be replaced.

5.2.1: COMPLETE OVERVIEW

The complete overview of the project has been given in the below Fig 5.1 and 5.2

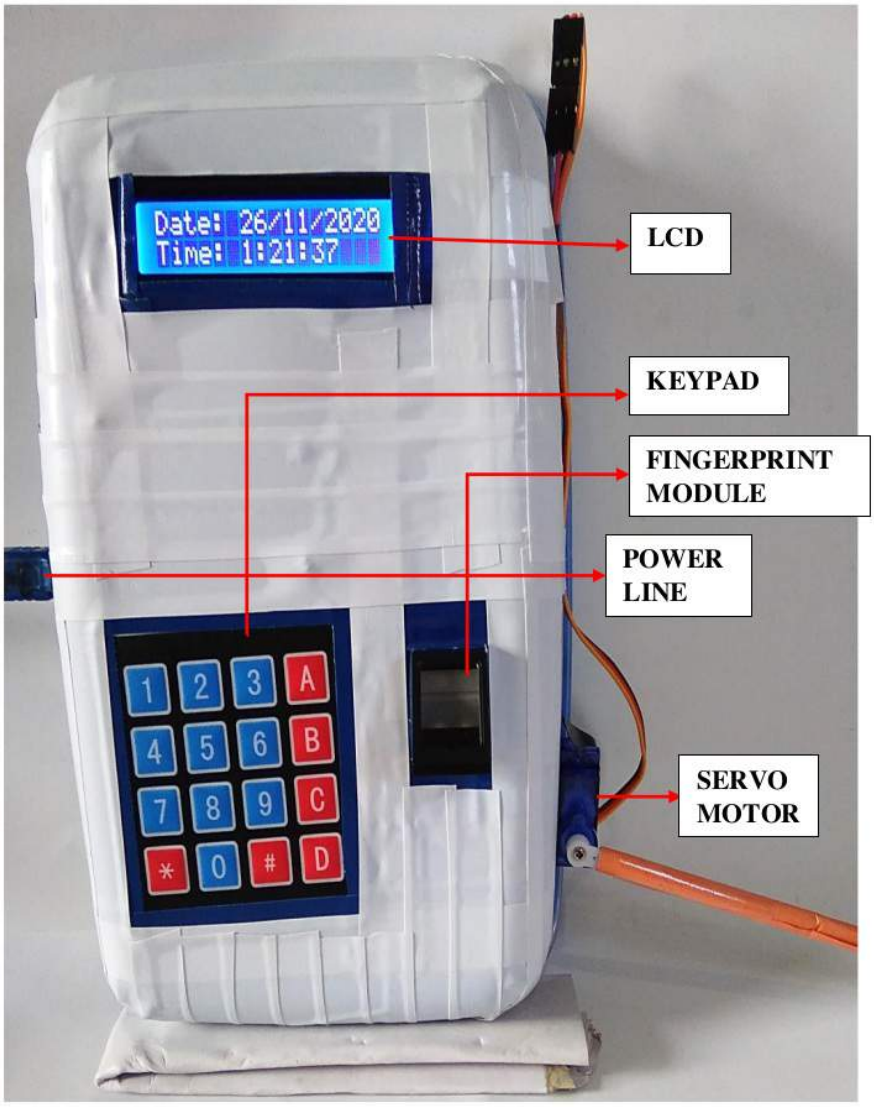


Fig 5.1: Front view of the system prototype.

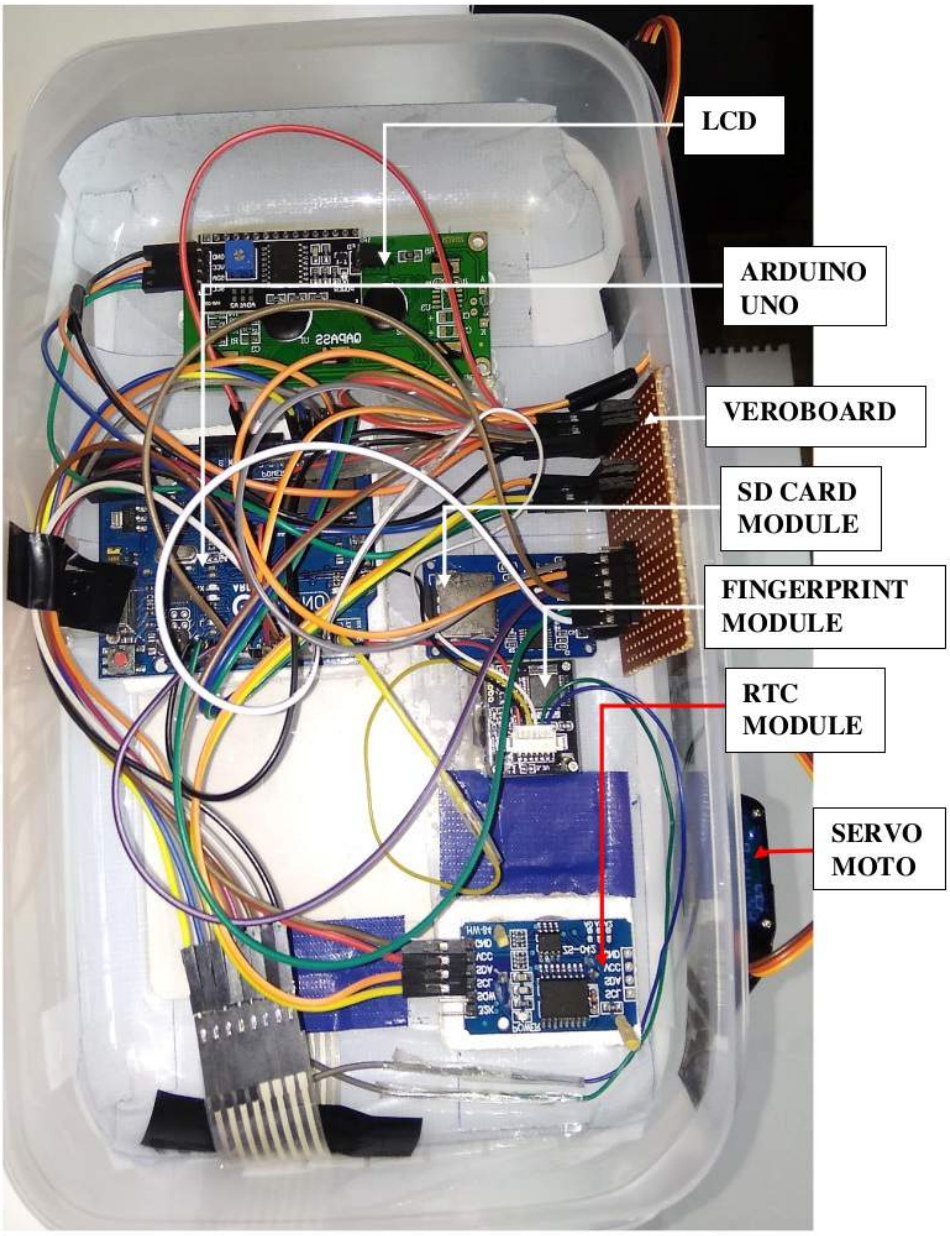


Fig 5.2: Back view of the system prototype.

5.3: PERFORMANCE OF SYSTEM

The system by default set to show time and date once the device is turned on. At first the device initialize the SD card module in order to save the data. Then it shows to press the button 'B' of keypad to take meal. If the user is registered with respective ID, it will allow to take meal by opening the gate but there is a condition that the user only take one meal at one time. Actually we designed it in that way no one will be allowed by the system to take meal second time until the meal array cleared by administrator in case of both lunch and dinner by pressing the button 'C' of keypad that open a window to enter a specific administrative password. If the user isn't registered, there is an option to be registered that is pressing the button 'A' of keypad that open a window to enter a specific administrative password. If the password is correct then there is option to enter respective ID which will be three digit number up to 127. After that there some instruction will be shown to register the fingerprint. If someone deny to take meal or close meal account or authority want to close someone's meal account there is an option to delete the fingerprint from device by pressing the button 'D' of keypad that open a window to enter a specific password. If the password it correct allow to enter ID which will be deleted. We have three different password in three different stage for security.

For Arduino UNO we need 5V DC and by supplying to it we are able to run the Fingerprint Module, LCD, SD card module, Servo motor and RTC module. In here the 5V for Arduino is provided from computer using USB cable or 5V adapter.

5.4: DEMONSTRATION AND RESULT OF THE PROJECT

Here we have shown the result that we have obtained from our project:

a: Initializing SD card.

b: After initialization.

c: Showing press B to take meal.

d: Place finger on fingerprint module.

Fig 5.3: Initialization of the system.

When the device is switched on, firstly its initialize the SD card module to save data with corresponding action has been shown in above Fig 5.3 (a and b).

After initialize the SD card, the device is ready to serve. In Fig 5.3 (c) it shows to press button 'B' on keypad in order to take meal.

When user pressed button 'B', it instructs to place finger on Fingerprint module. It shows in Fig 5.3 (d).

After placing finger, it replies 'Fingerprint Not Found' because initially no one has been registered which shows below in Fig 5.4.

Fig 5.4: Fingerprint doesn't found.

For being registered fingerprint user have to press button 'A' on keypad. Then it requires an administrative password to allow. If the password is correct, it allow to register. Otherwise it doesn't allow. It shows below in Fig 5.5 (a and b).

a: Require to enter the password.

b: If password is correct, then allow to register.

Fig 5.5: Fingerprint registration option.

The process of registering the fingerprint with respective ID shows below in Fig 5.6 (a, b, c, d, e and f).

At first, it requires a specific ID to register the fingerprint. The ID should be 3 digit number within 1 to 127.

a: Entering specific ID.

After enter the ID, then it requires to place the finger on surface of fingerprint module.

b: Place finger on fingerprint module.

Then the module take the picture of fingerprint and also ask to remove the finger from the surface.

c: The module take picture of fingerprint.

d: After image taken, then remove the finger.

For registering the fingerprint, the module ask to place the same finger again on the surface of module. Before register the fingerprint, the module cross match the picture of fingerprint which already taken.

e: Then it requires place the same finger again.

If the cross matching is fine, then the fingerprint is store successfully to the microcontroller.

f: Then it compares fingerprint and store it.

Fig 5.6: Registering fingerprint.

Now pressing the button 'B' to take meal. When it matched then it allowed to operate the servo motor to open the gate and allowed to take meal.

After 5 seconds the gate is automatically closed and it shown how many meals are remaining. Showing the respective result below in Fig 5.7 (a, b, c, d and e).

a: To take meal, it matches fingerprint which allocated with ID:1.

b: Then it allows to take meal.

c: Then the gate will be opened to take meal.

d: After few seconds, gate will be closed.

e: It also showing the reaming meal of users.

Fig 5.7: After matching the fingerprint.

Whenever the device allow any user to take meal, it store the data in a SD card as a file named 'DATA.TXT'. It shows below in Fig 5.8

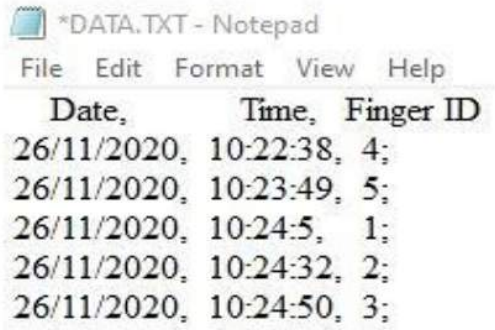


Fig 5.8: Data save at SD card.

If any user claim for meal though it already taken that time this will not allow the user to take meal until it cleared by system administrator it shown below in Fig 5.9

Fig 5.9: Meal already taken.

To clear the array of recording the meal taker, press the button ‘C’ on keypad. And here also an administrative password to continue the process. When the pressing password is correct then it will clear the array and it always done by authority. It shown below in Fig 5.10 (a, b and c).

a: Enter the password.

b: When password is correct.

c: Then the array will be cleared.

Fig 5.10: Clearing the array.

If any user or authority want to delete fingerprint with corresponding ID from this device, it need to press the button 'D' on keypad. For deleting ID here also enter the specific password. It shows below in Fig 5.11 (a, b, c and d).

a: Firstly ask to ensure of deleting fingerprint.

b: Enter the password.

c: Enter the specific ID.

d: Then it will deleted that ID.

Fig 5.11: Deleting the fingerprint.

5.5: APPROXIMATE COST OF THE PROJECT AND POWER ANALYSIS

This is really important to know the cost of project and how much power will consume by the project. So, total cost of our project and the power consumption are discussed below.

5.5.1: Total cost of the project

Our prototype project cost is around Tk.3710. If we want to produce for commercial purpose the cost can be reduced more by using only the chip instead of Arduino UNO Board and the components can be design on a single PCB board. So, in this case the cost will come around Tk.3200 which will be effective and all class of people can buy it However, a detailed feasibility study is compulsory for the analysis. Table 5.1 shows the total cost of the hardware of this project.

Table 5.1: Total cost of the project

No.	Component Name	Price (BDT)
1	Arduino UNO	380/-
2	R307 Fingerprint Module	2250/-
3	DS1307 RTC Module	130/-
4	SD Card Module	150/-
5	I2C 16x2 LCD	300/-
6	Servo Motor SG90	130/-
7	4x4 Matrix Keypad	70/-
8	Micro SD Card (4 GB)	200/-
9	Others	100/-
	Total:	**3710/-**

5.5.2: Total power consumption analysis

We calculate the total power consumption considering our system is 24/7 online. We take maximum ratings of all components. In real-time, this project's power consumption is less than the calculated values. A very deep feasibility study should be necessary for the most accurate result. In our expected real-world project, it will draw less power than that. In our project, from the official datasheet of all components, taking the maximum ratings on the consideration, our prototype project rating is 1.86625 Watt. To run the device, we need 5V DC supply which we have given through USB cable and battery but in case of real application we can use adapter or 5V battery or power bank type system. Table 5.2 shows the power consumption of this project.

Table 5.2: Power consumption of the project.

No.	Component Name	Required Voltage (V)	Operating Current (mA)	Power Consumption (Watt)
1	Arduino UNO	5	40	0.2
2	R307 Fingerprint Module	5	50	0.25
3	DS1307 RTC Module	5	1.5	0.0075
4	SD Card Module	5	30	0.15
5	I2C 16x2 LCD	5	1.75	0.00875
6	Servo Motor SG90	5	220	1.1
7	4x4 Matrix Keypad	5	30	0.15
			Total Power Consumption:	**1.86625**

CHAPTER 6

CONCLUSION

6.1: CONCLUSION

Distributing food in a cafeteria where hundreds of students come to eat lunch and dinner is a very difficult job. At the lunch time, maintaining the huge pressure with short stuff leads to chaotic situation. Students have to wait a long time for manual system of verification and sometimes gets late for their classes or works after lunch. Some also tries to get food without providing money. Our project will replace this manual system and make a more convenient and faster way to distribute food. It will also help to decrease the fraudulency in the food distribution. All of this is achieved without costing a lot amount of money.

6.2: FUTURE IMPROVEMENT

Here we are using a small prototype of the system. This can be further improved in future. It can be embedded with University student database which can ensure more functionality like full student profile and more security. It will also reduce the maintaining cost. A full functional data server can be installed for better storage and security of the student data. Instead of servo motor, we can use solenoid door lock control. Here we use three different password which is fixed but it can be changeable in future. Many more functions can be added in this this system in future.

6.3: LIMITATIONS

Every system has its own limitations. Here's our:

- Security of biometric data is the main limitation of this system. As biometric data is very sensitive, its security has to be airtight. Otherwise misuse of this data can make this system vulnerable.
- Our system can only store 127 fingerprint in current state. To make this work in a large environment it has to be upgraded hugely.
- Another limitation of this system is cost. To make this system working authority has to invest money, which may be a problem.

REFERENCES

[1] Manoj Senthil, Praveen Raj, Navin Kumar, Narendran, "Food Management System Based on Fingerprint Authentication," International Journal of Scientific & Technology Reserch Volume 9, Issue 03, March 2020.

[2] Resham Shinde, Priyanka Thakare, Neha Dhomne, Sushmita Sarkar, " Design and Implementation of Digital dining in Restaurants using Android ", International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 1, January 2014.

[3] Stefan Poslad. Ubiquitous computing: smart devices, environments and interactions. John Wiley and Sons, Ltd, Queen Mary, University of London, UK, 2009.

[4] Lavina Mall and Nihal Shaikh. "Canteen Management System Using RFID Technology Based on Cloud Computing", International Journal of Engineering Sciences & Research Technology, 2017.

[5] Monik Shah, Shalin Shah, Mohd Danish Shaikh,Kaustubh Tiwari. "Canteen Automation System", International Research Journal of Engineering and Technology (IRJET), Volume 5 Issue 1, January 2018.

[6] Kirti Bhandge, Tejas Shinde, Dheeraj Ingale, Neeraj Solanki, Reshma Totare, " A Proposed System for Touchpad Based Food Ordering System Using Android Application", International Journal of Advanced Research in Computer Science & Technology (IJARCST), Volume 3, Issue 1, March 2015.

[7] El-Pro-Cus, "Arduino UNO R3 Board", [Online]. Available at https://www.elprocus.com/what-is-arduino-uno-r3-pin-diagram-specification-and-applications/. [Accessed: October 7, 2020]

[8] Maker Pro, "Common communication peripherals of Arduino Uno", [Online]. Available at https://maker.pro/arduino/tutorial/common-communication-peripherals-on-the-arduino-uart-i2c-and-spi/. [Accessed: October 10, 2020]

[9] All About Circuit, "The Universal Asynchronous Receiver/Transmitter (UART)", [Online]. Available at https://www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart/ [Accessed: October 10, 2020].

[10] Vishnu M Aiea, “Interfacing R307 Optical Fingerprint Scanner with Arduino”, [Online]. Available at https://www.vishnumaiea.in/projects/hardware/interfacing-r307-optical-fingerprint-scanner-with-arduino/ [Accessed: October 15, 2020].

[11] Components 101, “4x4 Matrix Keypad”, [Online]. Available at https://components101.com/misc/4x4-keypad-module-pinout-configuration-features-datasheet/ [Accessed: October 20, 2020].

[12] Circuit Basics, “Set up a keypad on an Arduino”, [Online]. Available at https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/. [Accessed: October 25, 2020].

[13] Electronics Hub, “Pin Description of DS1307 RTC”, [Online]. Available at https://www.electronicshub.org/arduino-real-time-clock-tutorial/. [Accessed: November 1, 2020].

[14] Make-It Ca, “I2C LCD Interfacing”, [Online]. Available at https://www.make-it.ca/i2c-lcd-display-on-arduino/. [Accessed: November 5, 2020].

[15] Last Minute Engineers, “Interfacing Micro SD Card Module with Arduino”, [Online]. Available at https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/. [Accessed: November 10, 2020].

[16] Components 101, “SG90 Servo Motor”, [Online]. Available at https://components101.com/servo-motor-basics-pinout-datasheet/ [Accessed: November 15, 2020].

[17] Last Minute Engineers, “Servo Motor works and Interface it with Arduino”, [Online]. Available at https://lastminuteengineers.com/servo-motor-arduino-tutorial/. [Accessed: November 20, 2020].

APPENDIX

(SOURCE CODE)

```
//Including necessary libraries
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include <SPI.h>
#include <SD.h>
File myFile;
#include <RTClib.h>
#define RTC 0x68
RTC_DS1307 rtc;                    //Naming the DS1307 module as 'rtc'
#include <Servo2.h>
Servo myServo;
LiquidCrystal_I2C lcd(0x3F,16,2);
SoftwareSerial mySerial(2, 3);     //Fingerprint sensor wiring Rx to pin 3, Tx to pin 2

const byte numRows= 4;
const byte numCols= 4;
const int chipSelect = 4;
char keypressed;
char password1[4]={'1','2','3','A'};     //Define password
char password2[4]={'1','0','0','A'};
char password3[4]={'A','0','0','B'};
char userpress[4];                 //Array to get the code from the user

byte Array[10];
byte z, p1, value=0, V, j=0, value1=0, value2=0, value3=0, value4=0, value5=0, value6=0, value7=0,
value8=0, value9=0, value10=0;
boolean stat=1;
```

```
//keymap defines the key pressed according to the row and columns just as appears on the keypad
char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
byte rowPins[numRows] = {14,15,16,17};          //Define the pin connection of keypad
byte colPins[numCols]= {8,7,6,5};

//initializes an instance of the Keypad class
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
uint8_t id;

void setup()
{
  Serial.begin(9600);                    //Define baud rate for UART communication
  finger.begin(57600);                   //Define baud rate for fingerprint module
  Wire.begin();
  rtc.begin();
  //Set the RTC module to the current date and time.
  rtc.adjust(DateTime(F(__DATE__),F(__TIME__)));
  myServo.attach(10);
  myServo.write(10);
  lcd.init();
  lcd.backlight();
  lcd.clear();
DateTime now = rtc.now();
lcd.setCursor(0,0);
lcd.print(F("Date: "));
lcd.print(now.day(), DEC);
lcd.print('/');
lcd.print(now.month(), DEC);
```

```
lcd.print('/');
lcd.print(now.year(), DEC);
lcd.print(" ");
lcd.setCursor(0,1);
lcd.print(F("Time: "));
lcd.print(now.hour(), DEC);
lcd.print(':');
lcd.print(now.minute(), DEC);
lcd.print(':');
lcd.print(now.second(), DEC);
lcd.print(" ");
delay(2000);
 // setup for the SD card
Serial.print(F("Initializing SD card..."));
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print(F("Initializing"));
  lcd.setCursor(3, 1);
  lcd.print(F(" SD card..."));
  delay(1000);
if(!SD.begin(4))
{
Serial.println(F("initialization failed!"));
  lcd.clear();
  lcd.setCursor(3, 0);
  lcd.print(F("Failed !"));
  lcd.setCursor(0, 1);
  lcd.print(F("Check Connection"));
  delay(1000);
return;
}
Serial.println(F("initialization done."));
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print(F("Initialization"));
  lcd.setCursor(5, 1);
```

```
 lcd.print(F("Done"));
 delay(1000);

myFile=SD.open("DATA.txt", FILE_WRITE);        //open file
// if the file opened ok, write to it:
if (myFile)
{
Serial.println(F("File opened ok"));
myFile.println(" Date, Time, Finger ID");            // print the headings for data
}
myFile.close();
lcd.clear();
 lcd.setCursor(4, 0);
 lcd.print(F("Welcome"));
 delay(700);
 lcd.clear();
 lcd.setCursor(3, 0);
 lcd.print(F("Biometric"));
 lcd.setCursor(1, 1);
 lcd.print(F("Dining System"));
 delay(1000);
 lcd1();
}
void loop()
{
 keypressed = myKeypad.getKey();              //waiting for press key
 switch (keypressed)
      {
       case NO_KEY:
          // nothing to do
       break;

      case 'A':
{
       z=0;
        lcd.clear();
```

```
lcd.setCursor(3,0);
lcd.print(F("Welcome To"));
lcd.setCursor(4,1);
lcd.print(F("REGISTER"));
delay(800);
lcd.clear();
lcd.print(F("Enter Password:"));
getPassword1();                          //match the password 1
if(z==4)
{
lcd.clear();
lcd.print(F("Valid Password"));
delay(800);
lcd.clear();
lcd.setCursor(2,0);
lcd.print(F("Enter ID:"));
id = readnumber();        //enter ID using keypad
if (id == 0)
{                                 // ID #0 not allowed
 return;
}
 while (!getFingerprintEnroll());                //call enroll or register fingerprint function
 EEPROM.update(finger.fingerID, p1);        //save the fingerprint template to eeprom
}
else
{
lcd.clear();
lcd.setCursor(0,0);
lcd.print(F("Invalid"));
delay(1000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(F(" Try Again ! "));
lcd.clear();
lcd1();
}
```

```
 }
 break;

 case 'B':
 {
 for(int i=0;i<30;i++)
{
lcd.clear();
lcd.print(F("Place Finger"));
delay(500);
int result=getFingerprintIDez();              //matching fingerprint
delay(50);
if(result>=0)
{
 value = EEPROM.read(id);

   V = finger.fingerID;
   stat = checkID(V);              //check the ID
   if (!stat)
{
 Array[j]=V;                    //after checking ID it allows to take Meal
 j++;

 if (finger.fingerID==1)            //account of User 1
 {
  EEPROM[value1]=1;             //make an array for User 1
  value1++;
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value1);
delay(1000);
lcd.clear();
lcd1();
if ((30-value1)==0)
```

```
{
  value1=0;
}
}
if (finger.fingerID==2)            //account of User 2
{
  EEPROM[value2]=2;           //make an array for User 2
  value2++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value2);
delay(1000);
lcd.clear();
lcd1();
if ((30-value2)==0)
{
  value2=0;
}
}
if (finger.fingerID==3)                //account of User 3
{
  EEPROM[value3]=3;                //make an array for User 3
  value3++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value3);
delay(1000);
lcd.clear();
lcd1();
if ((30-value3)==0)
{
  value3=0;
```

```
}
}
if (finger.fingerID==4)                    //account of User 4
{
  EEPROM[value4]=4;                    //make an array for User 4
  value4++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value4);
delay(1000);
lcd.clear();
lcd1();
if ((30-value4)==0)
{
  value4=0;
}
}
  if (finger.fingerID==5)                    //account of User 5
{
  EEPROM[value5]=5;                    //make an array for User 5
  value5++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value5);
delay(1000);
lcd.clear();
lcd1();
if ((30-value5)==0)
{
  value5=0;
}
}
```

```
if (finger.fingerID==6)                  //account of User 6
{
  EEPROM[value6]=6;                      //make an array for User 6
  value6++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value6);
delay(1000);
lcd.clear();
lcd1();
if ((30-value6)==0)
{
  value6=0;
}
}
if (finger.fingerID==7)                  //account of User 7
{
  EEPROM[value7]=7;                      //make an array for User 7
  value7++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value7);
delay(1000);
lcd.clear();
lcd1();
if ((30-value7)==0)
{
  value7=0;
}
}
```

```
if (finger.fingerID==8)                 //account of User 8
{
 EEPROM[value8]=8;                      //make an array for User 8
 value8++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value8);
delay(1000);
lcd.clear();
lcd1();
if ((30-value8)==0)
{
 value8=0;
}
}
 if (finger.fingerID==9)                //account of User 9
{
 EEPROM[value9]=9;                      //make an array for User 9
 value9++;
6
lcd.clear();
lcd.setCursor(1, 0);
lcd.print(F("Remaining Meal"));
lcd.setCursor(6, 1);
lcd.print(30-value9);
delay(1000);
lcd.clear();
lcd1();
if ((30-value9)==0)
{
 value9=0;
}
}
```

```
if (finger.fingerID==10)                //account of User 10
 {
  EEPROM[value10]=10;                   //make an array for User 10
  value10++;
 lcd.clear();
 lcd.setCursor(1, 0);
 lcd.print(F("Remaining Meal"));
 lcd.setCursor(6, 1);
 lcd.print(30-value10);
 delay(1000);
 lcd.clear();
 lcd1();
 if ((30-value10)==0)
 {
  value10=0;
 }
 }
}
return;
}
}
lcd1();
 }
 break;

case 'C':
 {
        z=0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print(F("Enter Password:"));
        getPassword2();                 //match the password 2
        if(z==4)
        {
        lcd.clear();
        lcd.print(F("Valid Password"));
```

```
        delay(700);

        memset(Array, 0, sizeof(Array));          //clear the array
        j=0;
        lcd.clear();
        lcd.print(F("Array is Clear"));
        delay(1000);
        lcd1();
        }
        else
        {
        lcd.clear();
        lcd.print(F("Invalid"));
        delay(1000);
        lcd1();
        }
  }
 break;

case'D':
  {
        z=0;
        lcd.clear();
        lcd.setCursor(2,0);
        lcd.print(F("Are You Sure"));
        lcd.setCursor(2,1);
        lcd.print(F("to Delete ?"));
        delay(800);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print(F("Enter Password:"));
        getPassword3();                           //match the password 3
        if(z==4)
        {
         lcd.clear();
```

```
          lcd.setCursor(0,0);
          lcd.print(F("Enter ID:"));
          id = readnumber();                    //enter ID

         if (id == 0)
          {
            return;
          }
          deleteFingerprint(id);              //call the function to delete ID
          lcd1();
          return;
        }
        else
          {
          lcd.clear();
          lcd.print(F("Invalid"));
          delay(1000);
          lcd1();
          }
  }
  break;

default:

break;
}
}

boolean checkID(int test)                  //Function of checking ID
{
 for (int x = 0; x < 10; x++)
 {
  if (Array[x] == test)
  {
   lcd.clear();
   lcd.setCursor(1, 0);
```

```
    lcd.print(F("Already Taken"));
    delay(1200);
    lcd1();
    return true;
  }
}
59
lcd.clear();
lcd.setCursor(0,0);
lcd.print(F("Finger ID: "));
lcd.setCursor(10,0);
lcd.print(finger.fingerID);
delay(1500);
lcd.clear();
lcd.setCursor(4,0);
lcd.print(F("Allowed"));
delay(1000);
myServo.write(110);                    //send the signal to servo motor for opening gate
77
lcd.clear();
lcd.setCursor(3,0);
lcd.print(F("Gate Open"));
lcd.setCursor(3,1);
lcd.print(F("Take Meal"));
  loggingTime();                                //Call the function of real time & date
89
  myFile = SD.open("DATA.txt", FILE_WRITE);          //open the file on sd card
  if (myFile)
   {
    Serial.println(F("open with success"));
    myFile.print(finger.fingerID);                  //write the ID on sd card
    myFile.println(";");
   }
  myFile.close();
delay(5000);
myServo.write(10);                    //send the signal to servo motor for closing gate
92
lcd.clear();
lcd.setCursor(1,0);
lcd.print(F("Gate Closed"));
```

```
  delay(1000);
  lcd1();
  return false;
}
void getPassword1()                    //Function of Password 1
{
   for (int a=0 ; a<4 ; a++)
   {
    userpress[a]= myKeypad.waitForKey();
    lcd.setCursor(a,1);
    lcd.print(F("*"));
    delay(500);
   }
   lcd.clear();
   for (int b=0 ; b<4 ; b++)
   {
    if(userpress[b]==password1[b])
     z++;
   }
}
void getPassword2()                    //Function of Password 2
{
   for (int a=0 ; a<4 ; a++)
   {
    userpress[a]= myKeypad.waitForKey();
    lcd.setCursor(a,1);
    lcd.print(F("*"));
    delay(500);
   }
   lcd.clear();
   for (int b=0 ; b<4 ; b++)
   {
    if(userpress[b]==password2[b])
    z++;
   }
}
```

```
void getPassword3()                          //Function of Password 3
{
  for (int a=0 ; a<4 ; a++)
  {
   userpress[a]= myKeypad.waitForKey();
   lcd.setCursor(a,1);
   lcd.print(F("*"));
   delay(500);
  }
  lcd.clear();
  for (int b=0 ; b<4 ; b++)
  {
   if(userpress[b]==password3[b])
   z++;
  }
}
void loggingTime()                           //Function of Real & Date
{
DateTime now = rtc.now();
myFile = SD.open("DATA.txt", FILE_WRITE);
if (myFile)
{
myFile.print(now.day(), DEC);
myFile.print('/');
myFile.print(now.month(), DEC);
myFile.print('/');
myFile.print(now.year(), DEC);
myFile.print(',');
myFile.print(now.hour(), DEC);
myFile.print(':');
myFile.print(now.minute(), DEC);
myFile.print(':');
myFile.print(now.second(), DEC);
myFile.print(",");
}
myFile.close();
```

```
delay(1000);
}
void lcd1()
{
        lcd.clear();
        lcd.setCursor(1, 0);
        lcd.print(F("Press 'B' For"));
        lcd.setCursor(6, 1);
        lcd.print(F("MEAL"));
}
uint8_t readnumber(void) {                    //Function of Enter ID
  uint8_t num = 0;
   while (num == 0) {
     char keey = myKeypad.waitForKey();
    int  num1 = keey-48;
        lcd.setCursor(0,1);
        lcd.print(num1);
        keey = myKeypad.waitForKey();
    int  num2 = keey-48;
        lcd.setCursor(1,1);
        lcd.print(num2);
        keey = myKeypad.waitForKey();
    int  num3 = keey-48;
        lcd.setCursor(2,1);
        lcd.print(num3);
        delay(1000);
        num=(num1*100)+(num2*10)+num3;
        keey=NO_KEY;
   }
  return num;
}

uint8_t getFingerprintEnroll()                //Function of enroll or register
{
int p = -1;
lcd.clear();
```

```
lcd.print(F("Finger ID:"));
lcd.print(id);
lcd.setCursor(0,1);
lcd.print(F("Place Finger"));
delay(500);
while (p != FINGERPRINT_OK)
{
  p = finger.getImage();                    //Function of Capturing Image
22
switch (p)
{
case FINGERPRINT_OK:
Serial.println(F("Image taken"));
lcd.clear();
lcd.print(F("Image taken"));
delay (500);
break;
case FINGERPRINT_NOFINGER:
Serial.println(F("No Finger"));
3
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println(F("Communication error"));
lcd.clear();
lcd.print(F("Comm Error"));
break;
default:
Serial.println(F("Unknown error"));
lcd.clear();
lcd.print(F("Unknown Error"));
break;
}
}
// OK success!
p = finger.image2Tz(1);                     //Function of Converting First Fingerprint Image to template
52
switch (p) {
case FINGERPRINT_OK:
Serial.println(F("Image converted"));
```

```
break;
22
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println(F("Communication error"));
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println(F("Could not find fingerprint features"));
return p;
case FINGERPRINT_INVALIDIMAGE:
Serial.println(F("Could not find fingerprint features"));
return p;
default:
Serial.println(F("Unknown error"));
return p;
}
Serial.println(F("Remove finger"));
lcd.clear();
lcd.print(F(" Remove Finger "));
delay(1000);
p = 0;
while (p != FINGERPRINT_NOFINGER)
{
p = finger.getImage();                          //Function of Capturing Image again
}
3
Serial.print(F("ID ")); Serial.println(id);
p = -1;
Serial.println(F("Place same finger again"));
lcd.clear();
lcd.print(F("  Place Same "));
lcd.setCursor(0,1);
lcd.print(F(" Finger Again "));
while (p != FINGERPRINT_OK)
{
  p = finger.getImage();
switch (p)
{
case FINGERPRINT_OK:
```

```
Serial.println(F("Image taken"));
break;
case FINGERPRINT_NOFINGER:
Serial.print(".");
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println(F("Communication error"));
break;
default:
Serial.println(F("Unknown error"));
return;
}
}
// OK success!
p = finger.image2Tz(2);                    //Function of Converting Second Fingerprint Image to template
switch (p)
{
  case FINGERPRINT_OK:
Serial.println(F("Image converted"));
break;
case FINGERPRINT_PACKETRECIEVEERR:
Serial.println(F("Communication error"));
return p;
case FINGERPRINT_FEATUREFAIL:
Serial.println("Could not find fingerprint features");
return p;
default:
Serial.println(F("Unknown error"));
return p;
}
// OK converted!
Serial.print(F("Creating model for #")); Serial.println(id);
p = finger.createModel();                          //Function of Creating Model of Fingerprint
if (p == FINGERPRINT_OK)
{
Serial.println(F("Prints matched!"));
```

```
}
else if (p == FINGERPRINT_PACKETRECIEVEERR)
{
Serial.println(F("Communication error"));
return p;
}
else if (p == FINGERPRINT_ENROLLMISMATCH)
{
Serial.println(F("Fingerprints did not match"));
lcd.clear();
lcd.print(F(" FP Not Matched"));
delay(1000);
lcd.clear();
lcd.print(F("   Try Again ! "));
delay(1000);
lcd1();
return p;
}
else
{
Serial.println(F("Unknown error"));
return p;
}
Serial.print(F("ID ")); Serial.println(id);
p1 = finger.storeModel(id);                    //Function of store of Model
if (p1 == FINGERPRINT_OK)
{
Serial.println(F("Stored!"));
lcd.clear();
lcd.print(F("Stored!"));
delay(1500);
   lcd1();
}
else if (p == FINGERPRINT_PACKETRECIEVEERR)
{
Serial.println(F("Communication error"));
```

```
return p;
}
else
{
Serial.println(F("Unknown error"));
return p;
}
}
int getFingerprintIDez()                    //Function of Matching  Fingerprint
{
delay(50);
uint8_t p = finger.getImage();
if (p != FINGERPRINT_OK)
return -1;
p = finger.image2Tz();
if (p != FINGERPRINT_OK)
return -1;
p = finger.fingerFastSearch();
if (p != FINGERPRINT_OK)
{
lcd.clear();
lcd.print(F("Finger Not Found"));
lcd.setCursor(3,1);
lcd.print(F("Try Later"));
delay(2000);
return -1;
}
// found a match!
Serial.print(F("Found ID #"));
Serial.print(finger.fingerID);
Serial.print("\t");
return finger.fingerID;
}
uint8_t deleteFingerprint(uint8_t id)                    //Function of Deleting Fingerprint
{
  uint8_t p = -1;
```

```
 p = finger.deleteModel(id);                    //Function of Deleting Fingerprint Model

 if (p == FINGERPRINT_OK) {
  EEPROM.write(id,0);
  Serial.println(F("Deleted!"));
lcd.clear();
lcd.setCursor(3,0);
lcd.print(F("ID: "));
lcd.setCursor(6,0);
lcd.print(id);
 delay(300);
 lcd.setCursor(2,1);
 lcd.print(F("Deleted"));
 delay(1500);
 }
else
{
Serial.print(F("Unknown error: 0x"));
lcd.clear();
lcd.print(F("Something Wrong"));
lcd.setCursor(0,1);
lcd.print(F("Try Again Later"));
delay(1500);
   lcd1();
}
}
```

ET161015_ET161034_MAK_Report_Updated

ORIGINALITY REPORT

30%	25%	7%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	maker.pro Internet Source	2%
2	lastminuteengineers.com Internet Source	2%
3	how2electronics.com Internet Source	2%
4	Submitted to University of Southern Mississippi Student Paper	2%
5	dspace.iiuc.ac.bd Internet Source	1%
6	pastebin.com Internet Source	1%
7	create.arduino.cc Internet Source	1%
8	adafruit.github.io Internet Source	1%
9	www.ijarcsms.com Internet Source	1%

10	www.allaboutcircuits.com Internet Source	1%
11	Submitted to Kingston University Student Paper	1%
12	sns.chonbuk.ac.kr Internet Source	<1%
13	Submitted to Arts, Sciences & Technology University In Lebanon Student Paper	<1%
14	Submitted to University of Wolverhampton Student Paper	<1%
15	Submitted to Rajarambapu Institute of Technology Student Paper	<1%
16	A. Arjun, P. Umar Ahamed, K. Shahul Hameed, M.P.M. Mohamed Ibraheem, J. Mohammed Azharudeen, A. Muhammad Saleh. "Experimental analysis of solar wiper and its power estimates", Materials Today: Proceedings, 2020 Publication	<1%
17	www.irjet.net Internet Source	<1%
18	docs.particle.io Internet Source	<1%

19	www.makerguides.com Internet Source	<1%
20	docplayer.net Internet Source	<1%
21	Submitted to University of the Highlands and Islands Millennium Institute Student Paper	<1%
22	Submitted to International Islamic University Malaysia Student Paper	<1%
23	Jun Wei, Xuan Wang, Roshan Lalintha Peiris, Yongsoon Choi et al. "CoDine", Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11, 2011 Publication	<1%
24	Submitted to Nottingham Trent University Student Paper	<1%
25	Submitted to University of Southampton Student Paper	<1%
26	Submitted to University of Teesside Student Paper	<1%
27	Submitted to PSB Academy (ACP eSolutions) Student Paper	<1%
28	Submitted to Ghana Technology University College	<1%

	Student Paper	
29	Submitted to Tezpur University Student Paper	<1%
30	paper.researchbib.com Internet Source	<1%
31	www.ijstr.org Internet Source	<1%
32	Submitted to Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur Student Paper	<1%
33	circuitdigest.com Internet Source	<1%
34	eprints.utm.my Internet Source	<1%
35	Submitted to The London College UCK Student Paper	<1%
36	dokumen.pub Internet Source	<1%
37	Nitika, Swati Kumari, Vijay Kumar, Ranjan Kumar Behera. "Solar Powered Smart Home Design with IoT", 2020 IEEE-HYDCON, 2020 Publication	<1%
38	www.learningaboutelectronics.com Internet Source	<1%

39	Submitted to B.V. B College of Engineering and Technology, Hubli Student Paper	<1%
40	scholar.lib.vt.edu Internet Source	<1%
41	studylib.net Internet Source	<1%
42	sieuthilinhkien.vn Internet Source	<1%
43	Submitted to National University of Ireland, Galway Student Paper	<1%
44	www.dfrobot.com Internet Source	<1%
45	www.ijert.org Internet Source	<1%
46	www.flyrobo.in Internet Source	<1%
47	www.ijcsmc.com Internet Source	<1%
48	www.tinkerkit.com Internet Source	<1%
49	Najuka Jagtap, Jagannath Wadgaonkar, Kalyani Bhole. "Smart wrist watch", 2016 IEEE Students'	<1%

	Conference on Electrical, Electronics and Computer Science (SCEECS), 2016 Publication	
50	Submitted to American University of the Middle East Student Paper	<1%
51	www.electronicshub.org Internet Source	<1%
52	Submitted to Athens Metropolitan College Student Paper	<1%
53	Submitted to Open University Malaysia Student Paper	<1%
54	Submitted to University of Lincoln Student Paper	<1%
55	repository.usu.ac.id Internet Source	<1%
56	www.ijraset.com Internet Source	<1%
57	iotdesignpro.com Internet Source	<1%
58	www.arduino.cc Internet Source	<1%
59	www.cooking-hacks.com Internet Source	<1%

60	archive.org Internet Source	<1%
61	gist.github.com Internet Source	<1%
62	Submitted to South Bank University Student Paper	<1%
63	www.elprocus.com Internet Source	<1%
64	www.dirisala.net Internet Source	<1%
65	electrosome.com Internet Source	<1%
66	Submitted to Sheffield Hallam University Student Paper	<1%
67	arduino.cc Internet Source	<1%
68	www.openhacks.com Internet Source	<1%
69	Submitted to University of Nottingham Student Paper	<1%
70	upcommons.upc.edu Internet Source	<1%
71	Prachi Patil, Akshay Narkhede, Ajita Chalke,	

	Harshali Kalaskar, Manita Rajput. "Real time automation of agricultural environment", International Conference for Convergence for Technology-2014, 2014 Publication	<1%
72	Submitted to University of Wales Swansea Student Paper	<1%
73	randomnerdtutorials.com Internet Source	<1%
74	Submitted to GL Bajaj Institute of Technology and Management Student Paper	<1%
75	www.slideshare.net Internet Source	<1%
76	Submitted to KYUNG HEE UNIVERSITY Student Paper	<1%
77	robocraft.ru Internet Source	<1%
78	Jeremy Blum. "Exploring Arduino®", Wiley, 2019 Publication	<1%
79	"Passcode based Circuit Interrupter", International Journal of Innovative Technology and Exploring Engineering, 2020 Publication	<1%

80	documents.mx Internet Source	<1%
81	Submitted to Institute of Technology Carlow Student Paper	<1%
82	Submitted to University of Sunderland Student Paper	<1%
83	www.oreilly.com Internet Source	<1%
84	nicuflorica.blogspot.com Internet Source	<1%
85	Submitted to University of Sheffield Student Paper	<1%
86	www.littlecraft.com.my Internet Source	<1%
87	www.hobbytronics.net Internet Source	<1%
88	www.researchgate.net Internet Source	<1%
89	Submitted to University of Portsmouth Student Paper	<1%
90	rfm.zftp.com Internet Source	<1%
91	www.syris.com Internet Source	

		<1%
92	insis.vse.cz Internet Source	<1%
93	cyaninfinite.com Internet Source	<1%
94	Submitted to University of Glamorgan Student Paper	<1%
95	basicarduinotutorial.blogspot.com Internet Source	<1%
96	kincaidarcade.com Internet Source	<1%
97	www.cplusplus.com Internet Source	<1%
98	arduhope01.blogspot.com Internet Source	<1%
99	Neil Cameron. "Electronics Projects with the ESP8266 and ESP32", Springer Science and Business Media LLC, 2021 Publication	<1%
100	www.softwarepreservation.com Internet Source	<1%
101	hallroad.org Internet Source	<1%

102	www.appliedsensor.com Internet Source	<1%
103	www.ijircce.com Internet Source	<1%
104	Submitted to University of Technology, Sydney Student Paper	<1%
105	Submitted to University of Greenwich Student Paper	<1%
106	www.mickpeterson.org Internet Source	<1%
107	www.manelsoft.com Internet Source	<1%
108	openaccess.city.ac.uk Internet Source	<1%
109	Submitted to Higher Education Commission Pakistan Student Paper	<1%
110	cldup.com Internet Source	<1%
111	eprints.qut.edu.au Internet Source	<1%
112	"ICT Analysis and Applications", Springer Science and Business Media LLC, 2021 Publication	<1%

113	www.coursehero.com Internet Source	<1%
114	Sumita Nainan, Akshay Ramesh, Vipul Gohil, Jaykumar Chaudhary. "Speech controlled Automobile with Three-Level Biometric Security System", 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), 2017 Publication	<1%
115	sunupradana.info Internet Source	<1%
116	www.cours-gratuit.com Internet Source	<1%
117	wiki.keyestudio.com Internet Source	<1%
118	Bhaskar Kumar Mishra, Bhawani Singh Choudhary, Tanmay Bakshi. "Touch based digital ordering system on Android using GSM and Bluetooth for restaurants", 2015 Annual IEEE India Conference (INDICON), 2015 Publication	<1%
119	Asma Abdul Ghani Al-Shargabi, Francois Siewe. "A multi-layer framework for quality of context in ubiquitous context-aware systems", International Journal of Pervasive Computing and Communications, 2018	<1%

	Publication	
120	Neil Cameron. "Arduino Applied", Springer Science and Business Media LLC, 2019 Publication	<1%
121	Submitted to Universiti Teknologi Malaysia Student Paper	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		