Helwan University
faculty Of Engineering
Communication Department

Protecting Public Transport from Coronavirus

Presented by:

Abdallah Abdelwahed Mahdy

Diaa Mohamed Ezzat

Nadeen Yehia Mohamed

Naira Tarek Mahmoud

Nourhan Sayed Mohamed

Supervisor:

Assoc. Prof. Fatty Salem

July 2021

Contents

Acknowledgment 7
Abstract 8
.1 Features of project 9
Chapter 1 INTRODUCTION 10
1. Problem Definition 10
2. Motivation and Objectives 10
.3 How can we do that? 11
4. System design 12
Chapter 2 13
Hardware 13
1. The Micro controller (Atmega-32) 14
2. Communication protocol 16
2.1. UART 18
2.2. SPI 21
2.3 I2C 24
3. The GSM Module (Sim 900) 27
4. GPS Module (Neo 6M) 28
5. Wi-Fi Module (ESP 8266) 29
6. RFID Module (RC522) 30
7. Memory Device (EEPROM) 31
8. LCD (16*2) 31
9. Multiplexer 4*1 32
10. Software Architecture 32
11. BCP design 33
Chapter 3 34
hardware Code Implementation 34
1. Functions 35
The used functions in DIO 35
The used Functions in I2C: 35
The used functions in SPI 36
The used functions in UART 36
.2 Hardware part. 37
Used functions in EEPROM. 37

Used functions in GPS module. 37
Functions used in GSM module. 38
LCD functions: 39
RFID functions 40
Wi-Fi module functions 41
3. The main program 41
Chapter 4 48
Database 48
1. Database Management System 48
2. Basic functions of DBMS 48
3. Database System Environment 49
4. DB Approaches Characteristics 50
4.1 Data Abstraction 50
4.2 Support of multiple views of the Data 50
4.3 Multiuser Processing 50
5. DB Actors 51
5.1 DB Administrators 51
5.2 DB Designers 51
5.3 End Users 51
5.4 System Analysts and Application Programmers 51
5.5 DBMS designers and implementers 51
5.6 Tool developers 51
5.7 Operators and maintenance personnel 51
6. General Description of the Relational Model 52
7. SQL 54
8. Data types 55
9. Basic search commands 55
10. Creation, Deletion, and modification of table 55
Chapter 5 60
The Mobile Application 60
.1 Advantages 60
2. Objectives 60
3. What is flutter? 62
4. Dart 62

5. Why we used flutter? 63
5. Application design 64
Chapter 6 70
Project Cost analysis 70
1. Fixed cost 70
2. Variable Cost 70
Chapter 7 72
application code implementation 72
1. The code of log-in screen 72
2. The code of registration screen 75
3. The code of main screen 78
4. The code of search screen 84
5. The main code 86

Table of Figures

Figure 1: The Micro controller (Atmega-32)....14
Figure 2: parallel and serial communication 17
Figure 3: first protocol is UART....18
Figure 4: UART specifications 19
Figure 5: UART Data Frame Format....20
Figure 6: second protocol is SPI....21
Figure 7: SPI Specifications....22
Figure 8: SPI Concept....23
Figure 9: last protocol is I2C 24
Figure 10: I2C Bus interface....25
Figure 11: I2C Advantages 26
Figure 12: I2C disadvantages....26
Figure 13: The GSM Module (Sim 900)....27
Figure 14: GPS Module (Neo 6M)....28
Figure 15: Wi-Fi Module (ESP 8266)....29
Figure 16: RFID Module (RC522) 30
Figure 17: Memory Device (EEPROM)....31
Figure 18: LCD (16*2) 31
Figure 19: Multiplexer 4*1 32
Figure 20: software architecture....32
Figure 21: PCB design 33
Figure 22: Database System Environment 50
Figure 23: Data Abstraction....51
Figure 24: relation model 52
Figure 25: SQL with two identical tuples....59
Figure 26: multiset SQL....60
Figure 27: logo of the app 64
Figure 28: log-in screen 64
Figure 29: registration screen....65
Figure 30: app main screen 66
Figure 31: search screen....69
Figure 33: buses routes 68
Figure 32: side menu....70
Figure 34: card information....69
Figure 35: QR code scanner....69

Table 1: the priority of each part in the hardware....72

ACKNOWLEDGMENT

Our project required a lot of guidance and assistance from many people, and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

In a try to express our gratitude to our Supervisor **Assoc. Prof. Fatty Salem** who directed us in each step in this project, we are thankful for her guidance, patience, and motivation.

We also would like to place on record our sense of gratitude to everyone who rendered continuous aid over the period of study. Thanks for all the efforts they did to provide us with all useful information.

ABSTRACT

One of the most harmful problems to the environment is the problem of congestion, which leads to the spread of many diseases in addition to theft and road accidents, which also negatively affect public transport.

Especially with the spread of coronavirus and the proximity of people to each other and the exchange of money from one person to another spreading the virus quickly, which causes a great danger to human life.

That is why we want to avoid traffic to reduce the spread of the virus as much as possible.

To reduce the spread of Corona viruses must reduce congestion and this can be in two ways.

- The first way outside the bus through an application for each user that allows him to know the dates of the buses and the number of empty places, so he does not have to go down in a distance and cause congestion.
- The second way inside the bus by controlling the number of people available to enter the bus by locking on each chair this lock only works by the work of the system.

In this way, we can reduce the infection as much as possible.

Therefore, we will prevent the physical dealing between one person to another by preventing physical dealing and electronic payment.

1. Features of project

- Limiting the circulation of currencies between individuals.
- Facilitating payment in public transport between individuals by converting cash into electronic currencies.
- Individuals can pay through smart card by RFID card passes on Reader then deducts the price of the trip or the QR code by mobile application.
- Enabling customers to know when the next bus will be available and how many empty chairs are there using mobile application. This method helps reducing overcrowding in bus stations.

CHAPTER 1
INTRODUCTION

1. Problem Definition

We have suffered a lot from overcrowding and the large number of people gathering in places of transportation and within transportation in general, and now we are suffering a lot because of the spreading virus Corona, which is one of the most dangerous viruses that the world has passed through, which struck many countries and caused the death of many people. One of the most common ways that this virus spreads is crowding and people getting very close to each other, so the problem has become not only in crowding, but also with the spread of this virus and other viruses transmitted by touch.

2. Motivation and Objectives

After going through this crisis because of this virus, the world tried greatly to overcome this spread in various fields, as the doctors tried to produce drugs and medicines that could eliminate this virus, and on the other hand, government decisions to close public places to limit this spread until we return to normal life and This is what prompted us to try to find a solution to limit the spread of this virus, and this was the main motive behind our constant thinking and contribution to reducing the death of innocent people.

We thought of a solution to reduce gatherings in public transportation; thus, we will significantly reduce the spread of this virus. We found that there are two main problems, which are congestion inside transportation and at

transportation waiting stations, and dealing with money, which causes the spread of this virus very large we have developed a solution to reduce this congestion and another solution to prevent dealing with cash, and the payment is either electronically or via a card.

3. How can we do that?

There are two basic solutions, the first is a hardware solution, which will make it easier for the passenger to pay via the card without any money dealings with the driver or with anyone with some services such as the passenger's connection to the Internet in transportation The other solution is through a mobile application that enables him to know the transportation times, the number of seats available in each bus, the routes for each continuation, the cost of the trip as well, and the arrival time of the transportation, which will limit gatherings at stations to wait for transportation, and thus will greatly reduce the spread of the Corona virus And it will facilitate the process of riding transportation and payment in general.

4. System design

System Consist of Two main parts:

1. The Hardware System That will be placed in the bus.

2. The Mobile Application that will be with the users.

Firstly, we will talk about the hardware will be used in our system.

To design the system, we needed the following components:

1.1 The Micro controller (Atmega-32).

1.2 GSM Module (Sim 900).

1.3 GPS Module (Neo 6M).

1.4 Wi-Fi Module (ESP 8266).

1.5 RFID Module (RC522).

1.6 Memory Device (EEPROM).

1.7 LCD (16*2).

1.8 Multiplexer (4*1 Mux).

We will discuss all these parts later in more details.

CHAPTER 2

HARDWARE

As we mentioned earlier, the project is divided into two parts, the first is an application and the second is hardware.

We did research until we reached the most important basic parts in the hardware part, and we found that they are as follows:

1.1 The Micro controller (Atmega-32).

1.2 GSM Module (Sim 900).

1.3 GPS Module (Neo 6M).

1.4 Wi-Fi Module (ESP 8266).

1.5 RFID Module (RC522).

1.6 Memory Device (EEPROM).

1.7 LCD (16*2).

1.8 Multiplexer (4*1 Mux).

We will first discuss the most important component of the existing components, which is (Atmega-32)

1. The Micro controller (Atmega-32)

The high-performance, low-power Microchip 8-bit AVR RISC (Reduced-instruction-set Computing)-based microcontroller combines 32 KB ISP flash memory with read-while-write capabilities, 1 KB EEPROM, 2 KB SRAM, 54/69 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for boundary-scan and on-chip debugging/programming, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a universal serial interface (USI) with start condition detector, an 8-channel 10-bit A/D converter, programmable watchdog timer with internal oscillator, SPI serial port, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.

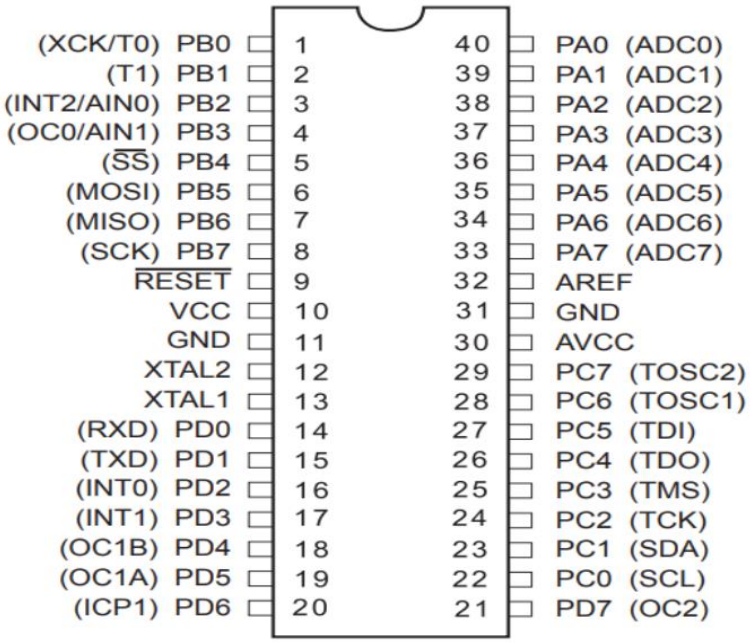


Figure 1: The Micro controller (Atmega-32)

We have decided to use The Atmega-32 for the following:

i. **Having the desired communication protocols.**

   The Atmega Have many communication protocols like Serial Peripheral Interface (SPI), Universal Asynchronous Receiver Transmitter (UART), Inter Integrated Circuit (I2C), And that what we need to interface our modules.

ii. **Wide compatibility with External Hardware to interface with.**

   The Atmega Have many pins to interface with external Peripherals it has 32 pin programmable as desired.

iii. **Various built-in peripherals.**

   It has built-In Peripherals like Digital Input/output (DIO), Timer To be able to run real time operations, Analog to Digital Converter (ADC) to be able to get readings from Sensors, Communication Protocols Like mentioned before, External Interrupts.

iv. **Perfect for prototypes**

   The Atmega-32 Have a fair price and a good performance so it is the best solution for the prototypes to start with.

v. **Have wide pin interfaces.**

   The Atmega-32 can handle the whole system without the need to Use another Controller as it has a 32-pin diagram so it can handle many modules.

Before talking about the rest of the components, we will talk about a very important topic, which is called Communication protocols.

2. Communication protocol

In complex systems, the functionality is divided into subsystems, each subsystem has microcontroller, and it is called ECU (Electronic Control Unit). These ECUs need to share the data between each other, i.e., they need to communicate with each other. So, the advantages of communication protocols are:

- Exchanging data between different subsystems within the same system
- Reduce the complexity of a system by splitting it into different subsystems.
- Transfer the data on different distances and on different mediums.

A protocol is a defined method of communication by defining two main aspects:

- Hardware Interface This activity defines the hardware connections (wires) between (ECUs).
- Data Frame Format this activity defines the data frame transmitted of the wires between the nodes including the number of bits and arrangement.

In our project we have three types of protocols:

- UART: Universal Asynchronous Receiver Transmitter.
- SPI: Serial Peripheral Interface.
- I2C: Inter-Integrated Circuit.

There is parallel and serial communication. Parallel communication sending multiple bits together and serial communication sending bit by bit.

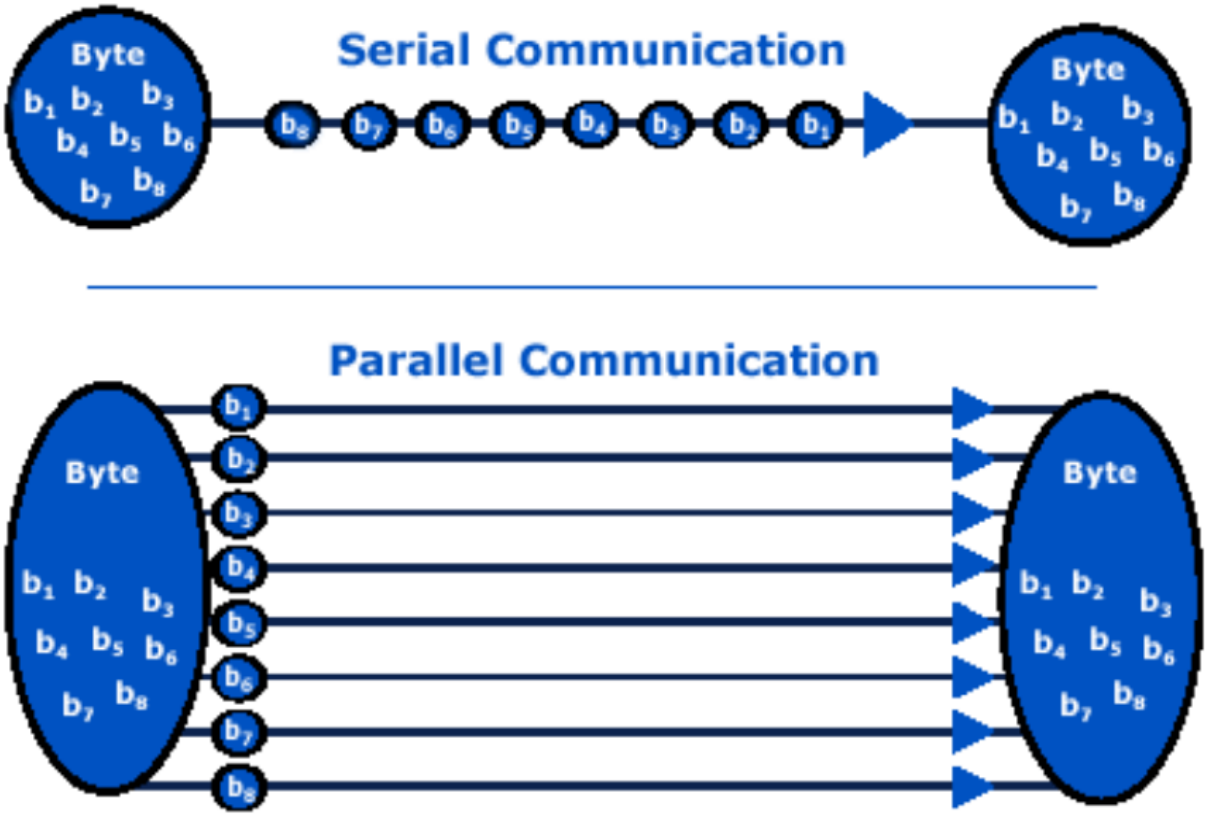


Figure 2: parallel and serial communication

And after we talked about the types of communication, we will now talk about each of the previous protocols.

2.1. UART

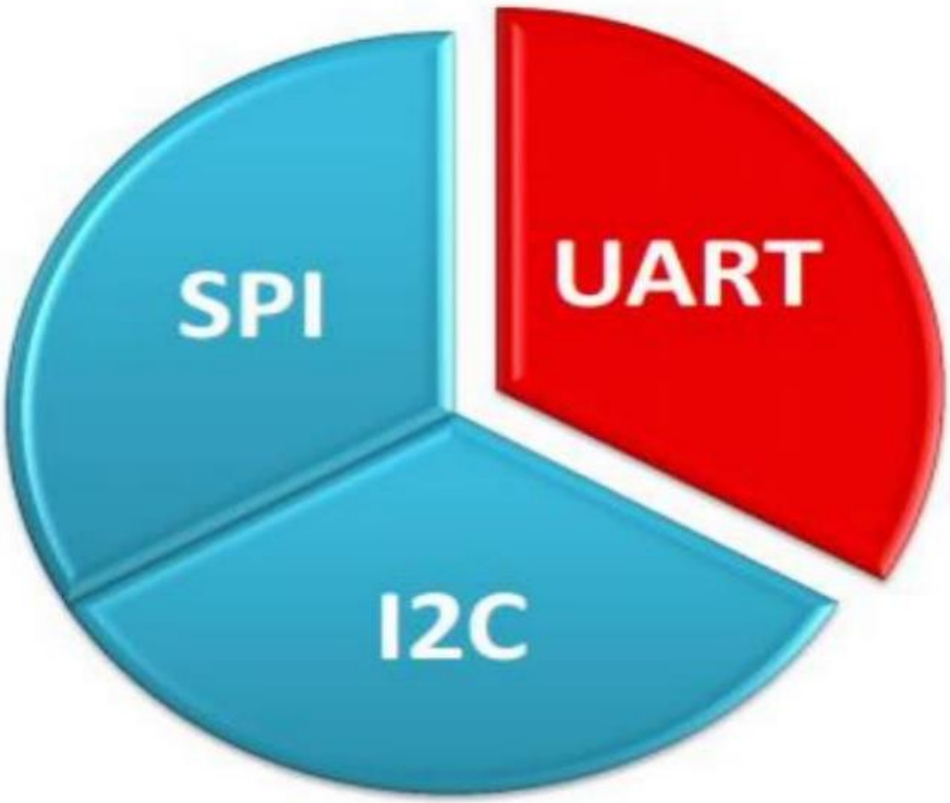


Figure 3: first protocol is UART

UART stands for Universal Asynchronous Receiver Transmitter. It is a serial communication protocol that consists of one wire for transmitting data and one wire to receive data. A common parameter is the baud rate known as "bps" which stands for bits per second. If a transmitter is configured with 9600bps, then the receiver must be listening on the other end at the same speed.

2.1.1 UART specifications

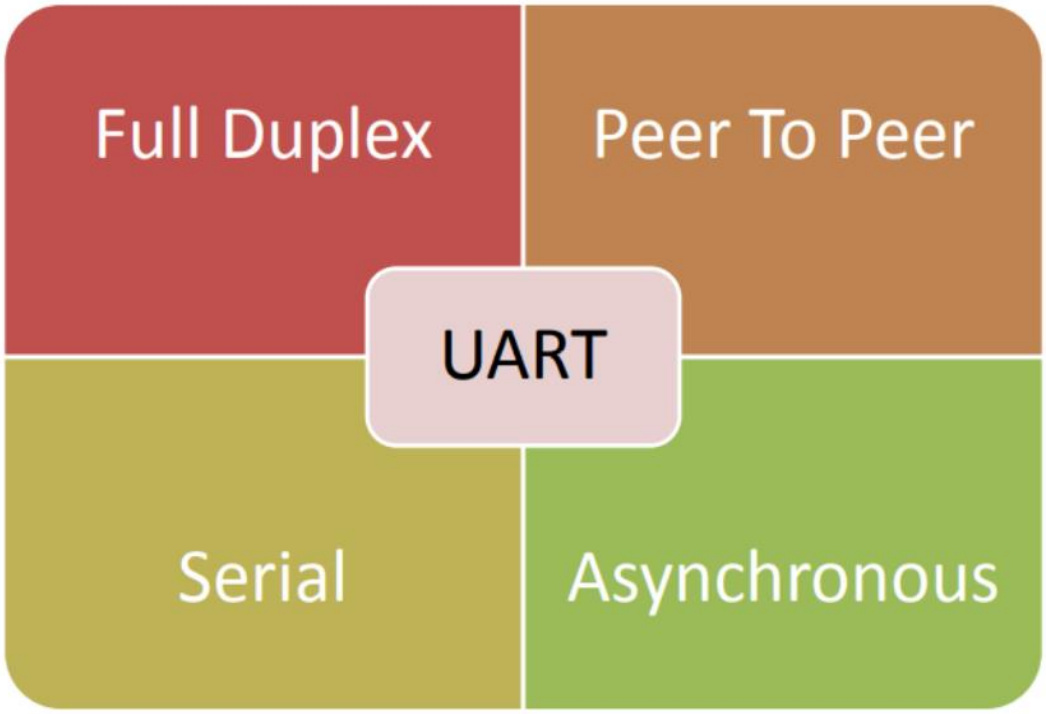
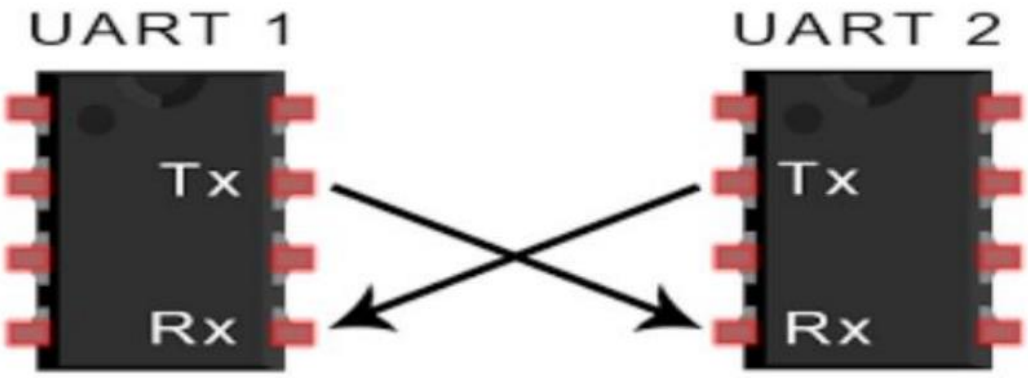


Figure 4: UART specifications

Each node has two lines called Tx (Transmission line) and Rx (Receive Line). The Tx of one node shall be connected to Rx of the other node and vice versa.



1.1.2 UART Data Frame Format

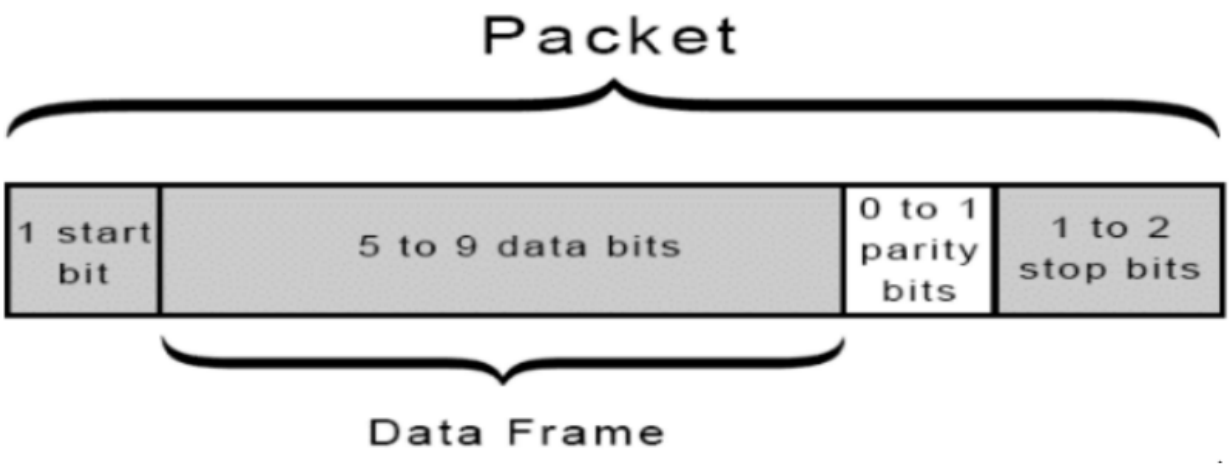


Figure 5: UART Data Frame Format

- Start bit: 1 bit indicates the start of a new frame, always logical low.
- Data: 5 to 9 bits of sent data.
- Parity bit: 1 bit for error checking
  1. Even parity: clear parity bit if number of 1s sent is even.
  2. Odd parity: clear parity bit if number of 1s sent is odd.
- Stop bit: 1 or 2 bits indicate end of frame, always logic high.

2.2. SPI

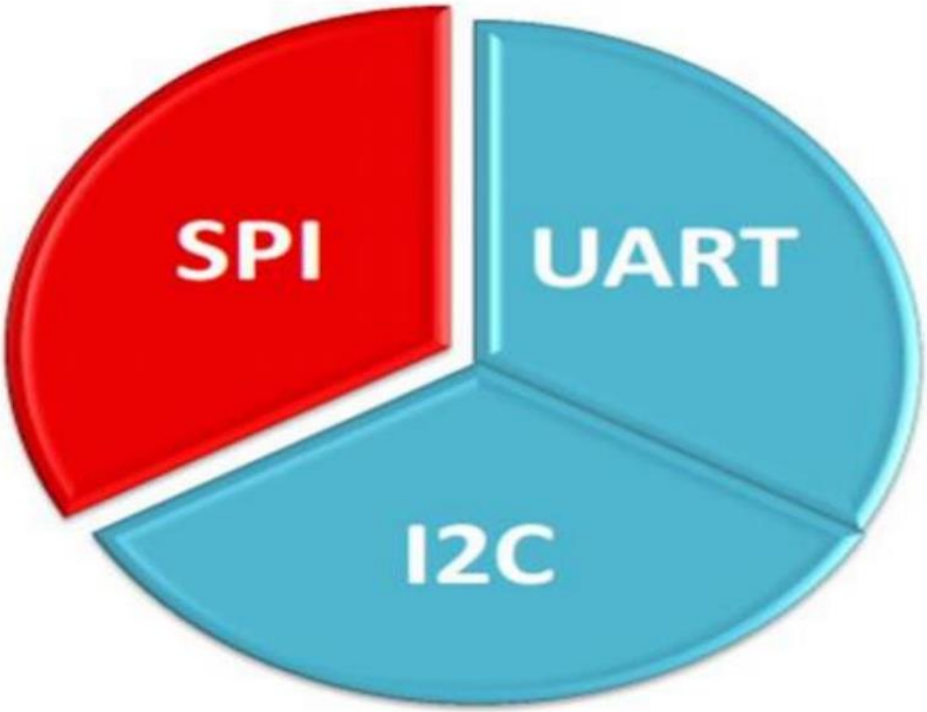


Figure 6: second protocol is SPI

Serial Peripheral Interface (SPI) bus is a synchronous serial communication interface specification used for short distance communication.

• The SPI bus can operate with a single master device and with one or more slave devices.

2.2.1 Master Slave Communication

In this type of communication there is a master node that can send data to any other nodes (Slaves). The master is the only node that can initiate the communication; the slave can never initiate the communication. The 10 slaves can send data to master only when the master permit the slave to send.

The Master / Slave network can be divided to:

- Single Master Single Slave (SMSS)
- Single Master Multi Slave (SMMS)
- Multi Master Multi Slave (MMMS)

2.2.2 SPI Specifications

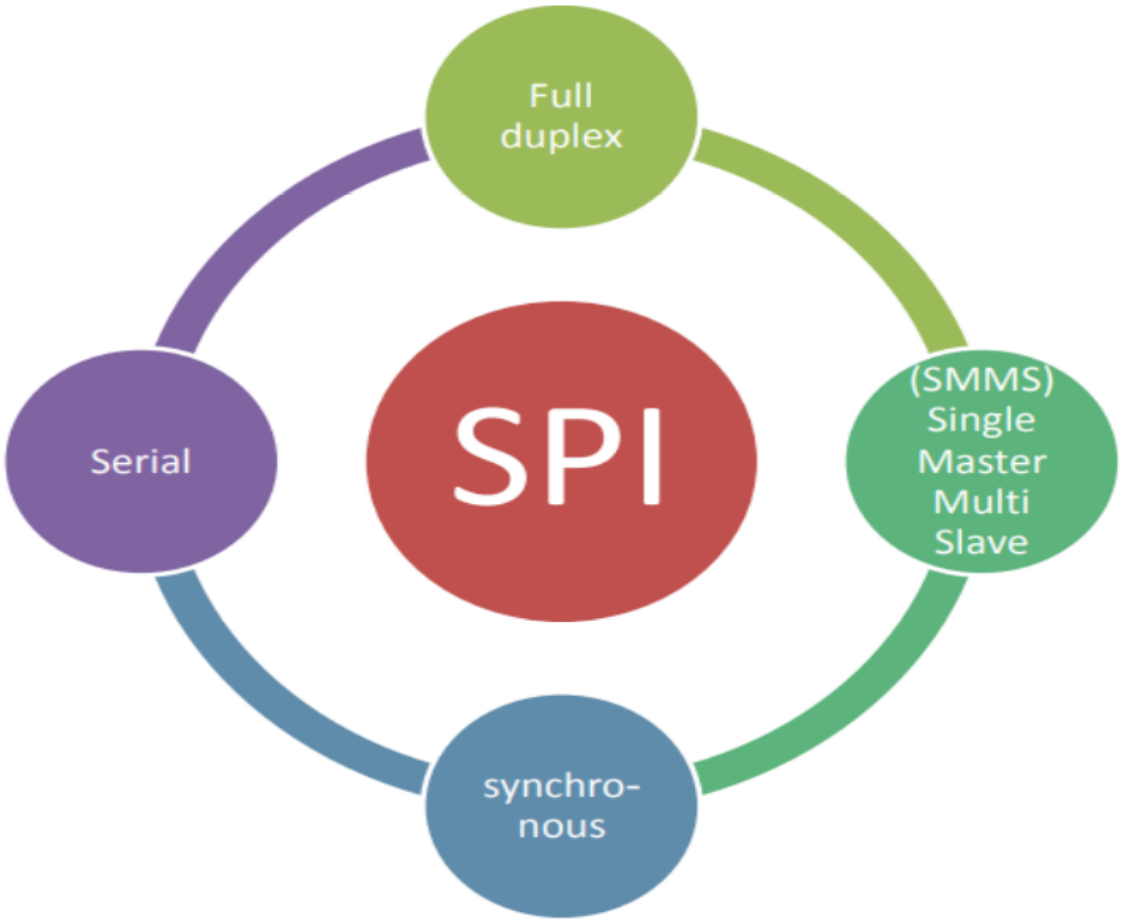


Figure 7: SPI Specifications

2.2.3 SPI connections

- SCLK: Serial Clock (output from master).
- MOSI: Master Output, Slave Input (output from master).
- MISO: Master Input, Slave Output (output from slave).
- SS: Slave Select (active low, output from master).

2.2.4 SPI concepts

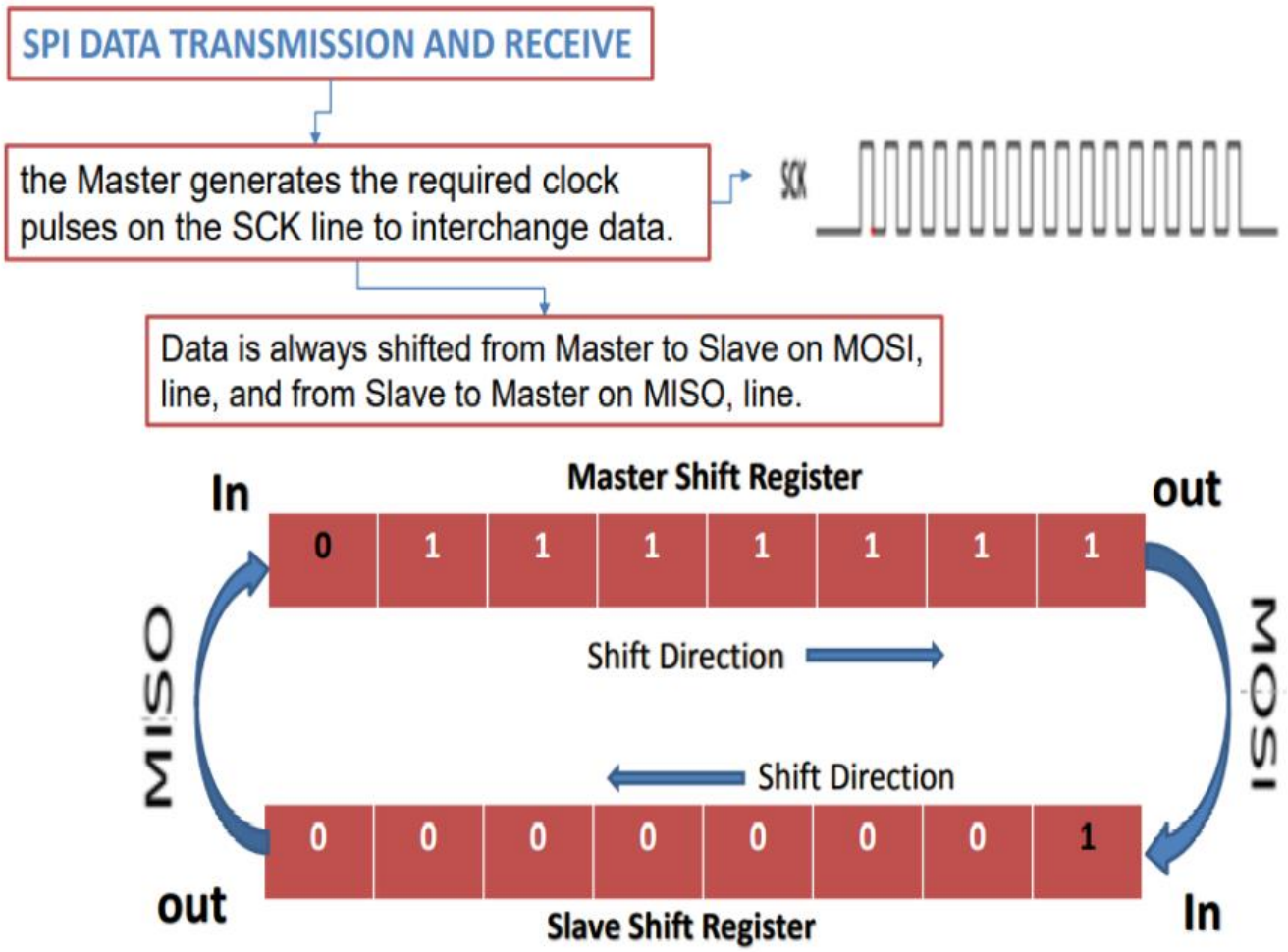


Figure 8: SPI Concept

2.3 I2C

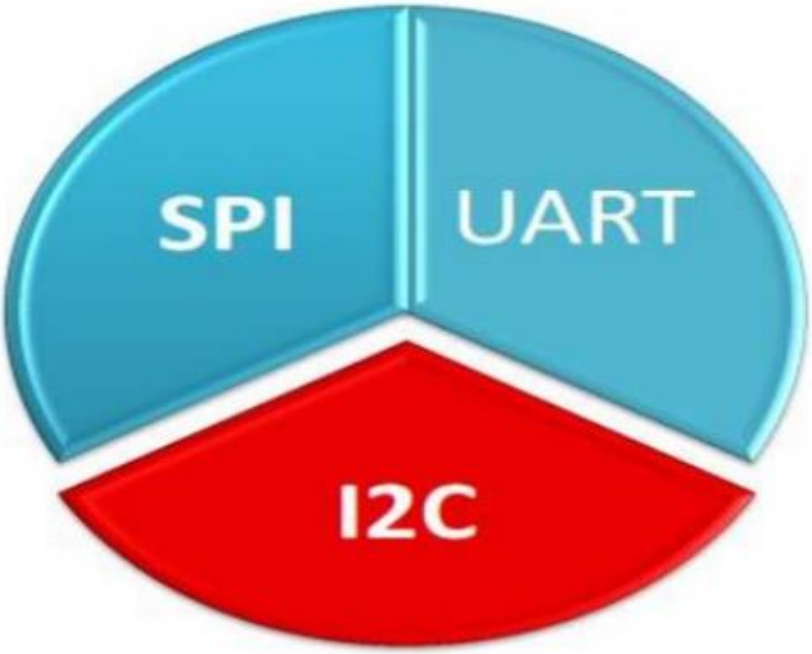


Figure 9: last protocol is I2C

Inter-Integrated Circuit I²C (Pronounced I Two C or I Squared C), is a serial communication protocol at which the devices are hooked up to the I2C bus with just two wires.

• It is sometimes referred to as Two Wire Interface or the TWI.

• Devices could be the CPU, IO Peripherals like ADC, or any other device which supports the I2C protocol.

• All the devices connected to the bus are classified as either being Master or Slave.

2.3.1 I2C Bus interface

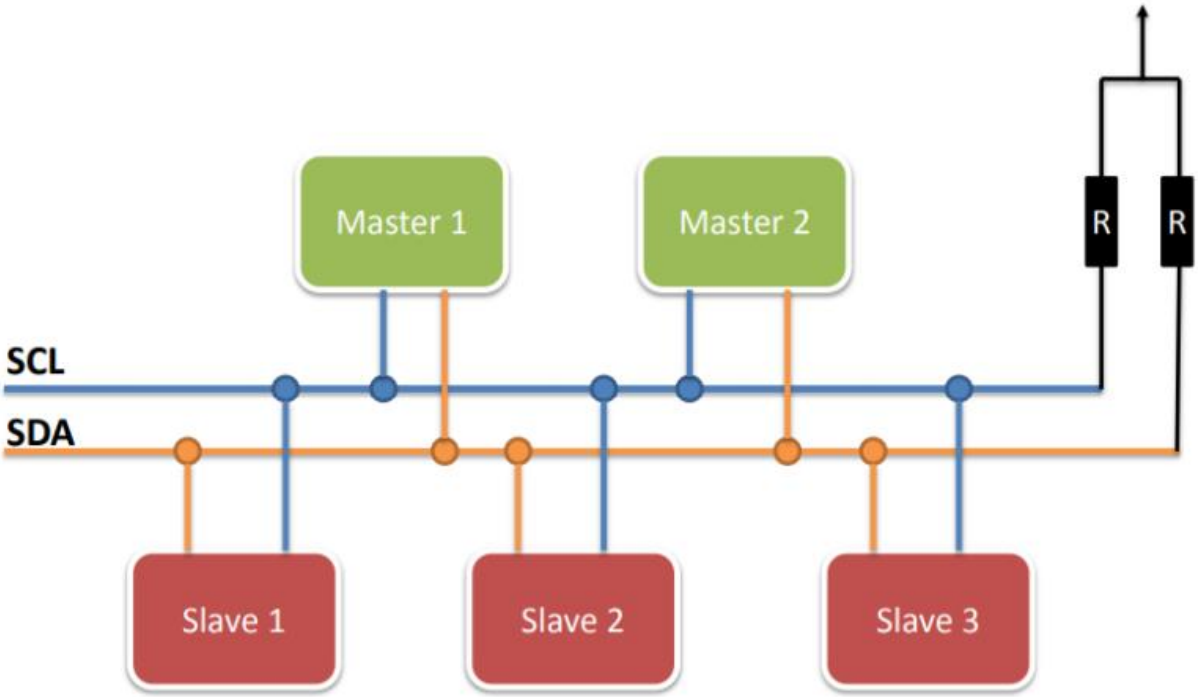


Figure 10: I2C Bus interface

2.3.2 Serial Data Line (SDA)

The Serial Data Line (SDA) is the data line. All the data transfer among the devices takes place through this line. Every Byte put on SDA must be 8 bit long. Each Byte followed by Acknowledge bit by the receiver.

2.3.3 Serial Clock Line (SCL)

Is the serial clock. I2C is a synchronous protocol, and hence, SCL is used to synchronize all the devices and the data transfer together. The active master is the responsible for driving this line. When SCL is low- Data can be transfer "to avoid the start and stop conditions".

2.3.4 I2C Advantages

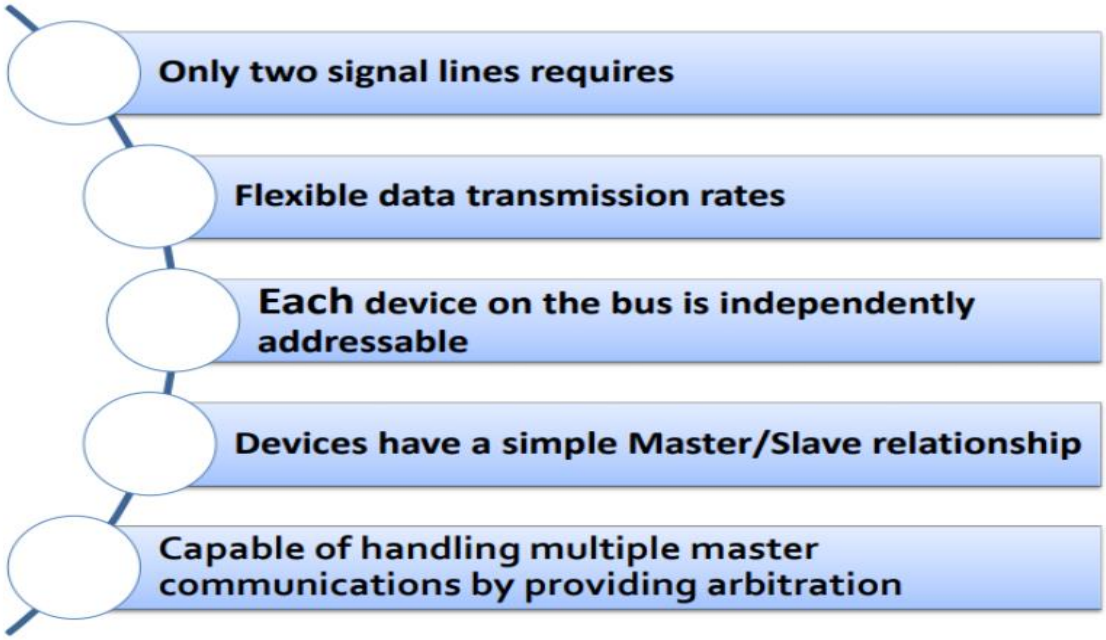


Figure 11: I2C Advantages

2.3.5 I2C Disadvantages

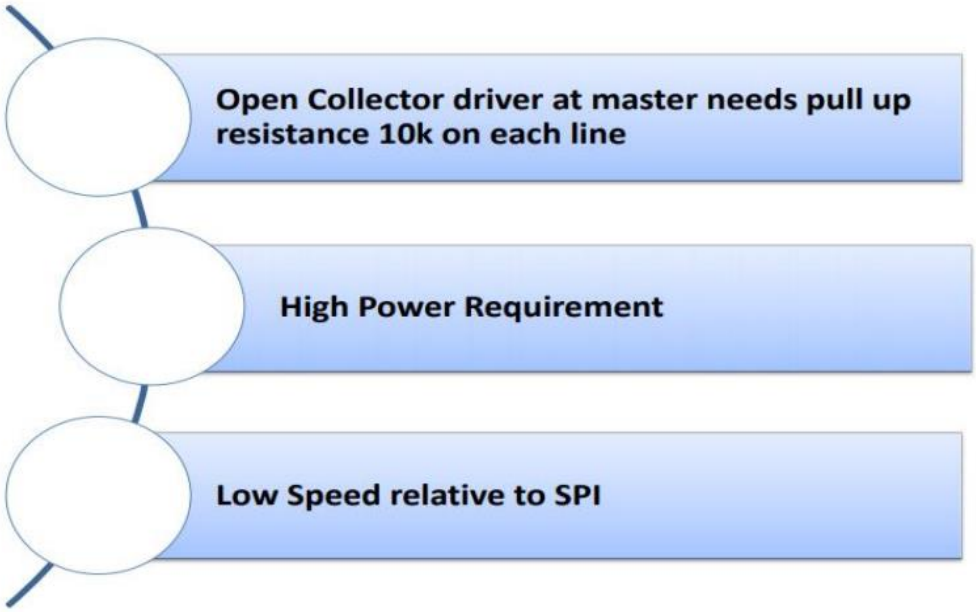


Figure 12: I2C disadvantages

After we talked about the protocols, we will talk about the rest of the components that interact with the controller using these protocols

3. The GSM Module (Sim 900)

The GSM module is device that transfers the data from the user side to the data base; it has a sim card that uses the mobile network to connect the system to the internet service. It uses UART communication protocol to communicate with the microcontroller. The GSM and the microcontroller agree on a baud rate (19200) that the two devices communicate with; the microcontroller sends special commands called at command that the GSM can understand and respond to it.

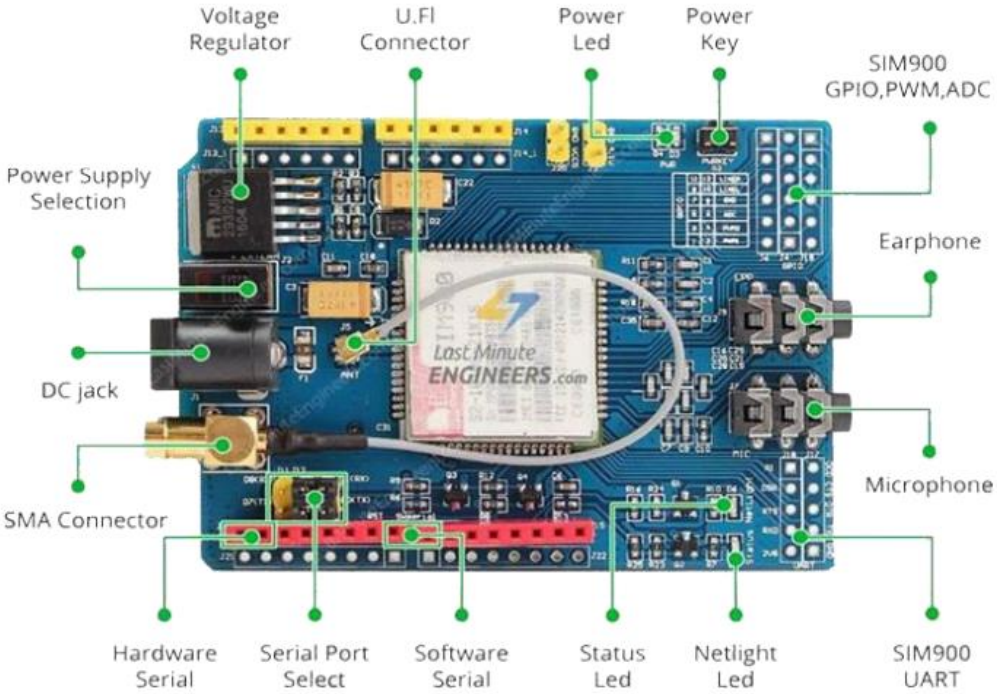


Figure 13: The GSM Module (Sim 900)

To Interface the Sim 900 with the micro controller we need to initiate some functions and set configurations in the Atmega 32 (will be discussed in the next chapter)

The GSM has many modes to operate with like:

- Receive a message.
- Send a message.
- Establish a call.
- Receive a call.

4. GPS Module (Neo 6M)

GPS module contains processors and antennas that receive data sent by satellites through dedicated RF frequencies. it will receive timestamp from each visible satellite, along with other pieces of data. It provides the time and location data of the buses to the system data base to calculate the estimated time and other data needed. It uses UART communication protocol to interface with the micro controller.

Figure 14: GPS Module (Neo 6M)

5. Wi-Fi Module (ESP 8266)

It is a module that will help us to create a Wi-Fi network for the application users as it is not needed for them to have a data network plan to be able to use our system, we will provide the network connectivity for them.

It uses UART communication protocol to communicate with the microcontroller.

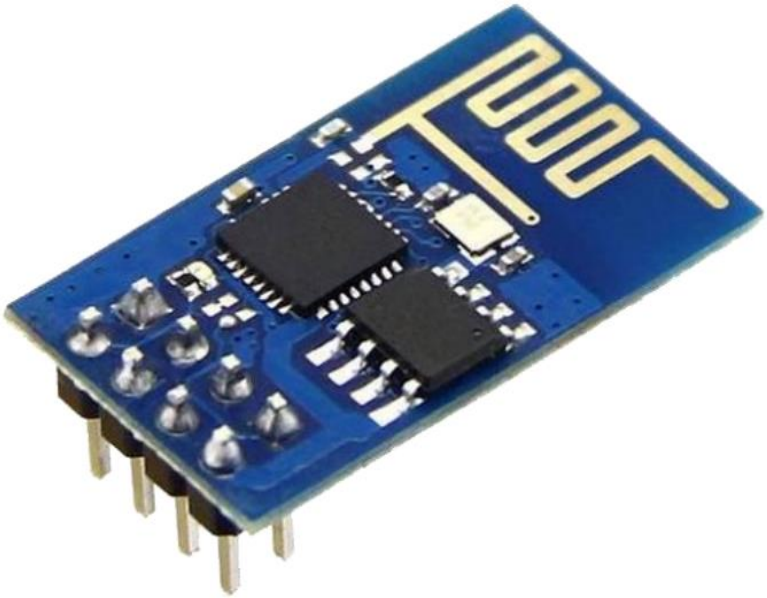


Figure 15: Wi-Fi Module (ESP 8266)

6. RFID Module (RC522)

It is the way to authenticate the Smart Card Users, it consists of two main parts:

- The reader which consists of frequency reading device and communicate with the microcontroller With SPI (Serial Peripheral Interface) which can provide a high-speed Communication between the controller and the reader.
- The Smart Card which can hold a data like Id and card serial Number which identifies the users.

Figure 16: RFID Module (RC522)

7. Memory Device (EEPROM)

- It stores the temporary copy of User data until it then transmitted to the database.
- It uses I2C (Inter Integrated Circuit) Communication Protocol to communicate with the Microcontroller.

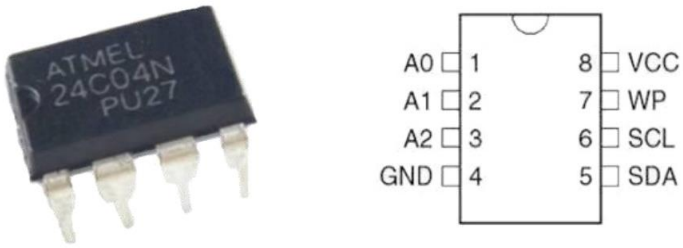


Figure 17: Memory Device (EEPROM)

8. LCD (16*2)

It is used to display the data to the users to indicate when the process is done and display the necessary data. It can display 32 characters at the same time. It uses 8 pins to transfer data and 3 pins to control the display.

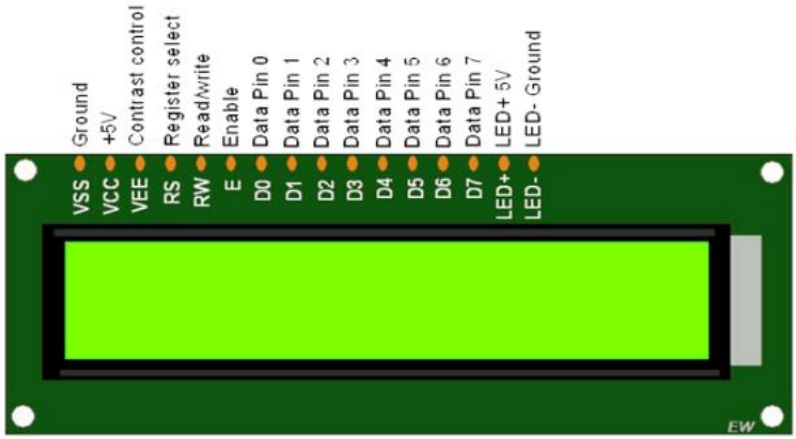


*Figure 18: LCD (16*2)*

9. Multiplexer 4*1

We will need the Mux to be able to attach the three modules that uses UART to the controller to be able to select one of them at a time. It has 4 pins as input, two selection lines and one output; the two selection lines make you decide which input will be connected to the output.

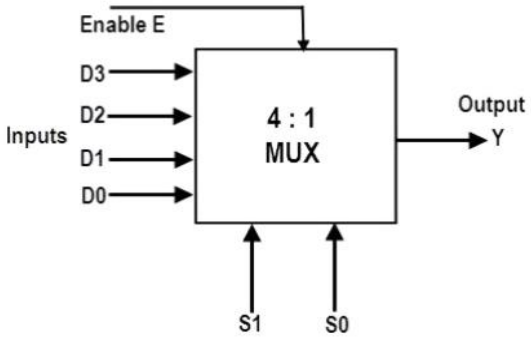


*Figure 19: Multiplexer 4*1*

10. Software Architecture

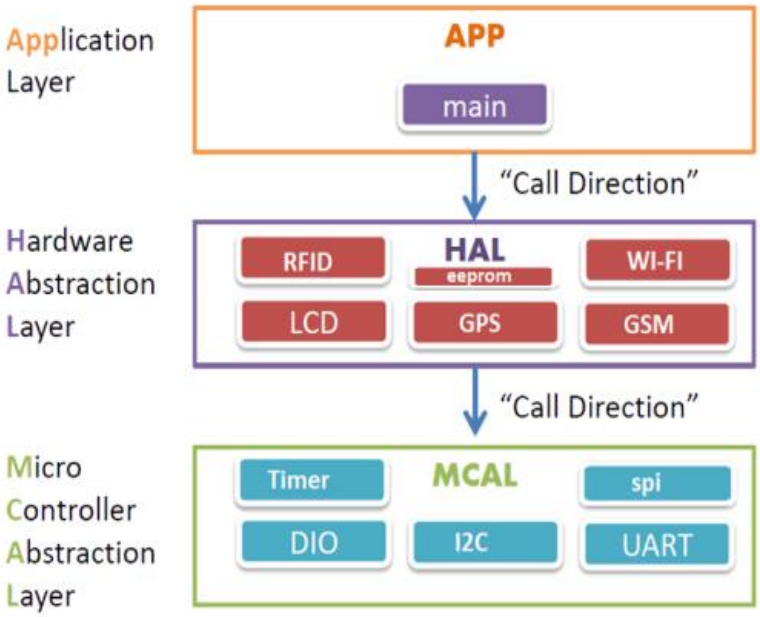


Figure 20: software architecture

11. BCP design

We have designed the hardware circuit diagram using “proteus 8 professional”, this program allows you to place all the hardware component and start connecting it manually in schematic capture.

Then, it will move you to PCB layout, you start placing your copper board edges and start placing the components you have used in schematic.

You will find a feature called “auto routing” that will begin to draw the circuit for you, after that you will have the layout and start preparing it to form it on the copper board and start placing real components.

The PCB layout of this project is as following:

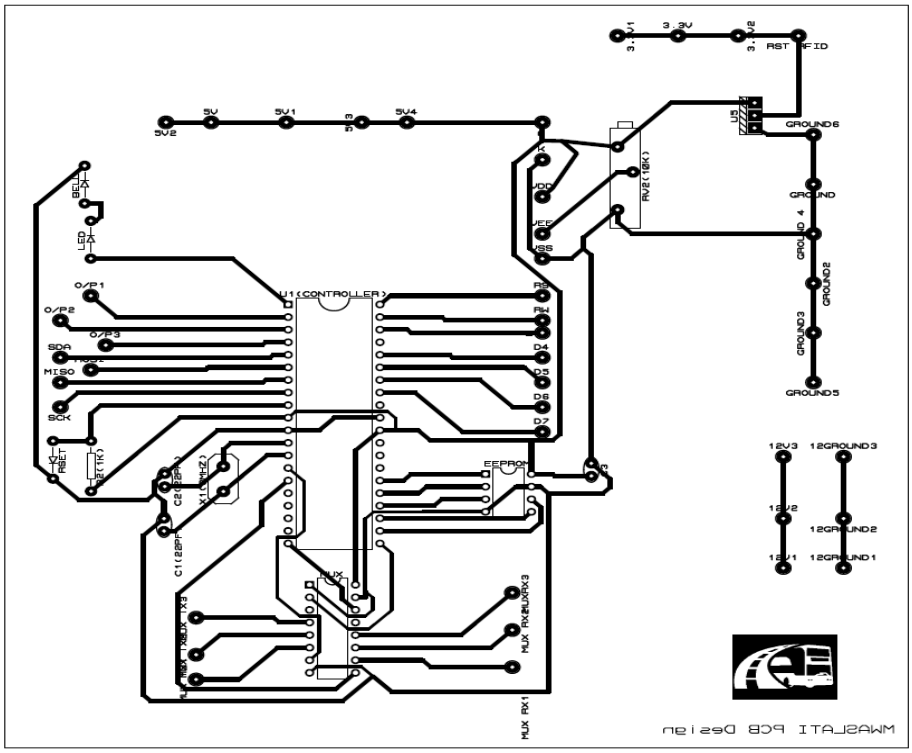


Figure 21: PCB design

CHAPTER 3

HARDWARE CODE IMPLEMENTATION

The hardware part is implemented using Embedded system C programming, it is implemented using AVR microprograming chip ATMEGA32.

We will explain the used code and the functions we have used in this code in this chapter:

Microcontrollers have variable peripherals that it can deal with, we will use some of these peripherals in our project like:

- DIO.
- I2C.
- SPI.
- UART.

And it will interface with hardware to implement its functions like:

- LCD Module.
- EEPROM Module.
- GSM Module.
- GPS Module.
- Wi-Fi Module.
- RFID Module.
- The Multiplexer.

1. Functions

We will discuss the used functions in each one just the functions names

The used functions in DIO

void DIOSetPortDirection (u8 port number, u8 value);

- used to set the port if it is input or output.

void DIOSetPortValue (u8 port number, u8 value);

- used to set the port value if it is high or low.

void DIOSetPinDirection(u8 portnumber,u8 Pinnumber,u8 value);

- used to set only one pin direction if it is input or output.

void DIOSetPinValue (u8 port number, u8 Pin number, u8 value);

- used to set a pin value if it is high or low.

u8 **DIOGetPinValue** (u8 port number, u8 Pin number);

- used to read any pin value if this pin is input.

The used Functions in I2C:

void i2c_init_master(**void**);

- used to initialize peripheral as master communication.

void i2c_init_slave(**void**);

- used to initialize peripheral as slave communication.

void i2c_start(**void**);

- used to start the transmission.

void i2c_repeated_start(**void**);

- to reestablish the communication.

void i2c_send_slave_address_with_write_req(**unsigned char** slave_address);

- used to send slave address with write request.

void i2c_send_slave_address_with_read_req(unsigned char slave_address);

- used to send slave address with read request.

void i2c_write_byte(unsigned char byte);

- used to write a byte.

unsigned char i2c_read_byte(void);

- used to read a byte.

void i2c_stop(void);

- used to stop communication.

voidi2c_slave_check_slave_address_received_with_write_req (void);

- used to check slave address with write request.

voidi2c_slave_check_slave_address_received_with_read_req(void);

- used to check slave address with read request.

unsigned char i2c_slave_read_byte(void);

- used to let the slave to read data.

void i2c_slave_write_byte(unsigned char byte);

- used to let the slave to write data.

The used functions in SPI

void spi_init();

- Used to initialize the SPI communication.

uint8_t **spi_transmit**(uint8_t data);

- Used To Transmit data in SPI.

The used functions in UART.

void USART_Init(unsigned long);

- UART initialization function.

char USART_RxChar();

- Data receive function.

void USART_TxChar(**char**);

- Data transmit function.

void USART_SendString(**char***);

- Used to send array of data.

2. Hardware part.

Used functions in EEPROM.

void EEpromInit(**void**);

- Used to initialize the module.

void EEpromWriteByte(**unsigned short** address, **unsigned char** data);

- Used to write data to an address in the memory.

unsigned char EEpromReadByte(**unsigned short** address);

- Used to read data from an address in the memory.

Used functions in GPS module.

void convert_time_to_UTC();

- Used to get time from the GPS module.

void convert_to_degrees(**char** *);

- Used to get the location into degree.

void get_gpstime();

- Function to get the real-time from the gps module.

void get_latitude(uint16_t);

- Function to get Latitude data from gps module.

void get_longitude(uint16_t);

- Function to get longitude data from gps module.

void get_altitude(uint16_t);

- Function to get altitude data from gps module.

Functions used in GSM module.

void Read_Response();

- Function to read the GSM response.

void Start_Read_Response();

- Function to start the response.

void Buffer_Flush();

- Function to empty all the buffers.

void GetResponseBody(**char***, uint16_t);

- Function to get the response body.

bool **WaitForExpectedResponse**(**char***);

- Boolean function to check for response.

bool **SendATandExpectResponse**(**char***, **char***);

- Function to check for AT commands.

bool **HTTP_Parameter**(**char***, **char***);

- Function to check for web parameters.

bool **SIM900HTTP_Start**();

- Ask for web service start.

bool **SIM900HTTP_Connect**(**char***, **char***, **char***);

- Ask for web connection.

bool **HTTP_Init**();

- Ask for web initialization.

bool **HTTP_Terminate**();

- Ask for web service termination.

bool **HTTP_SetURL**(**char** *);

- Set URL for website and ask for location.

bool **HTTP_Connected**();
- Ask for connection state.

bool **HTTP_SetPost_json**();
- Ask for data set state.

bool **HTTP_Save**();
- Ask if action is saved.

bool **HTTP_Data**(**char***);
- Post data action and ask for response.

bool **HTTP_Action**(**char**);
- Give action and make sure that it is sent.

bool **HTTP_Read**(uint8_t, uint16_t);
- Read data from the website.

uint8_t **HTTP_Post**(**char*** , uint16_t);
- Post data to the server.

uint8_t **HTTP_get**(**char** *, uint16_t);
- Get data from server.

bool **SIM900HTTP_Init**();

- Establish connection and make sure that it is established.

LCD functions:

void LCDInit(uint8_t style);
- Initialize the LCD.

void LCDWriteString(**const char** *msg);
- Write string to LCD.

void LCDWriteInt(**int** val,**unsigned int** field_length);
- Write int to LCD

```
void LCDGotoXY(uint8_t x,uint8_t y);
```

- Set pointer to a position on lcd.

```
void LCDHexDumpXY(uint8_t x, uint8_t y,uint8_t d);
```

- Display Hex data on LCD.

```
void LCDByte(uint8_t,uint8_t);
#define LCDCmd(c) (LCDByte(c,0))
#define LCDData(d) (LCDByte(d,1))
```

- Operate the LCD on 4-bit mode.

```
void LCDBusyLoop();
```

- This function waits till LCD is busy.

RFID functions

```
void mfrc522_init();
```

- Initialize the reader.

```
void mfrc522_reset();
```

- Reset the reader to its initial state.

```
void mfrc522_write(uint8_t reg, uint8_t data);
```

- Write data to RFID reader.

```
uint8_t mfrc522_read(uint8_t reg);
```

- Read date from the reader.

```
uint8_t     mfrc522_request(uint8_t req_mode, uint8_t * tag_type);
```

- Request mode from the reader.

```
uint8_tmfrc522_to_card(uint8_tcmd,uint8_t*send_data,
uint8_tsend_data_len,uint8_t*back_data,uint32_t *back_data_len);
```

- Write data to card.

Wi-Fi module functions

```
void ESP_Read_Response(char*);
void ESP8266_Clear();
void ESP_Start_Read_Response(char*);
void ESP_GetResponseBody(char*, uint16_t);
bool ESP_WaitForExpectedResponse(char*);
bool ESP_SendATandExpectResponse(char*, char*);
bool ESP8266_ApplicationMode(uint8_t);
bool ESP8266_ConnectionMode(uint8_t);
bool ESP8266_Begin();
bool ESP8266_Close();
bool ESP8266_WIFIMode(uint8_t);
uint8_t ESP8266_JoinAccessPoint(char*, char*);
uint8_t ESP8266_connected();
uint8_t ESP8266_Start(uint8_t, char*, char*);
uint8_t ESP8266_Send(char*);
int16_t ESP8266_DataAvailable();
uint8_t ESP8266_DataRead();
uint16_t Read_Data(char*);
```

- It is like GSM module functions

3. The main program

```
#define F_CPU 8000000UL
#define _NOP()  asm("nop")
#define SREG    _SFR_IO8(0x3f)
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "USART.h"
#include "lcd.h"
#include "utils.h"
#include "spi.h"
#include "mfrc522.h"
```

```
#include "I2c.h"
#include "eeprom.h"
#include "std_types.h"
#include "gsm.h"
#include "wifi.h"
#include "gps.h"
```

- Include all the needed files.

```
#define gsm        1
#define gps        2
#define wifi       3

int active=1;

extern int8_t Response_Status, CRLF_COUNT ;
extern uint16_t Counter;
 extern uint32_t TimeOut;
extern char RESPONSE_BUFFER[DEFAULT_BUFFER_SIZE];
extern int8_t Response_Status1;
extern volatile int16_t Counter1, pointer;
extern uint32_t TimeOut1;
extern char RESPONSE_BUFFER1[DEFAULT_BUFFER_SIZE1];

extern char
Latitude_Buffer[15],Longitude_Buffer[15],Time_Buffer[15],Altitude_Buffer[8]
;
extern char degrees_buffer[degrees_buffer_size]; /* save latitude or longitude
in degree */
extern char GGA_Buffer[Buffer_Size]; /* save GGA string */
extern uint8_t GGA_Pointers[20]; /* to store instances of ',' */
extern char GGA_CODE[3];
extern volatile uint16_t GGA_Index, CommaCounter;

extern bool IsItGGAString ,flag1,flag2 ;

uint8_t SelfTestBuffer[64];
```

- Initialize all the needed global variables.

```
void activate(int n)
{
        DIOSetPinDirection(3,6,1);
```

```
        DIOSetPinDirection(3,7,1);
        if(n==1){
                USART_Init(19200);
                DIOSetPinValue(3,6,0);
                DIOSetPinValue(3,7,0);
                active=1;
        }
        if(n==2){
                        USART_Init(9600);
                        DIOSetPinValue(3,6,1);
                        DIOSetPinValue(3,7,0);
                        active=2;
                }
        if(n==3){
                        USART_Init(115200);
                        DIOSetPinValue(3,6,0);
                        DIOSetPinValue(3,7,1);
                        active=3;
                }
}
```

- Function to select the active UART Module.

```
int main()
{
char _buffer[150];
GGA_Index=0;
DIOSetPinDirection(1,0,1);
DIOSetPinDirection(1,1,1);
DIOSetPinDirection(1,2,1);
DIOSetPinDirection(1,2,1);
DIOSetPinDirection(3,6,1);
DIOSetPinDirection(3,7,1);
uint8_t data;
uint8_t byte;
uint8_t test=0,test1=0;
uint8_t str[MAX_LEN];
uint8_t check[MAX_LEN]={0x42,0x49,0x91,0x1E,0x84,0xFB,0x8C,0xBB};
uint8_t check1[MAX_LEN]={0xE9,0x08,0x9A,0x6E,0x15,0xFB,0x8c,0xBB};
LCDWriteStringXY(2,0,"initializing System");
_delay_ms(1000);
LCDClear();
LCDInit(LS_BLINK);
LCDWriteStringXY(2,0,"RFID Reader");
LCDWriteStringXY(5,1,VERSION_STR);
```

```
spi_init();
_delay_ms(1000);
LCDClear();

//init reader
mfrc522_init();

//check version of the reader
byte = mfrc522_read(VersionReg);
if(byte == 0x92)
{
	LCDWriteStringXY(2,0,"MIFARE RC522v2");
	LCDWriteStringXY(4,1,"Detected");
}else if(byte == 0x91 || byte==0x90)
{
	LCDWriteStringXY(2,0,"MIFARE RC522v1");
	LCDWriteStringXY(4,1,"Detected");
}else
{
	LCDWriteStringXY(0,0,"No reader found");
}

byte = mfrc522_read(ComIEnReg);
mfrc522_write(ComIEnReg,byte|0x20);
byte = mfrc522_read(DivIEnReg);
mfrc522_write(DivIEnReg,byte|0x80);

_delay_ms(1500);


EEpromInit();
LCDClear();
activate(gsm);
LCDWriteStringXY(0,0,"Initializing GSM");
	USART_Init(19200);	/* Initiate
USART with 19200 baud rate */
	sei();	/* Start
global interrupt */
	while(!SIM900HTTP_Start());
	while(!(SIM900HTTP_Connect(APN, USERNAME, PASSWORD)));
	SIM900HTTP_Init();
```

```
activate(gps);


        memset(GGA_Buffer, 0, Buffer_Size);
        memset(degrees_buffer,0,degrees_buffer_size);

    //_delay_ms(3000);            /* wait for GPS receiver to initialize */
    USART_Init(9600);           /* initialize USART with 9600 baud rate
*/
        sei();

activate(wifi);

uint8_t Connect_Status;

        USART_Init(115200);                                  /* Initiate
USART with 115200 baud rate */
        sei();                                               /* Start
global interrupt */

        //while(!ESP8266_Begin());
ESP8266_WIFIMode(BOTH_STATION_AND_ACCESPOINT);/* 3 = Both
(AP and STA) */
ESP8266_ConnectionMode(SINGLE);                  /* 0 = Single; 1 =
Multi */
ESP8266_ApplicationMode(NORMAL);            /* 0 = Normal Mode; 1 =
Transperant Mode */
        if(ESP8266_connected() ==
ESP8266_NOT_CONNECTED_TO_AP)
        ESP8266_JoinAccessPoint(SSID, PASSWORD);
        ESP8266_Start(0, DOMAIN, port);
```

- Module initialization Part.

```
while(1){

        byte = mfrc522_request(PICC_REQALL,str);
        LCDHexDumpXY(0,1,byte);
        LCDClear();
        LCDWriteStringXY(2,0,"Please Scan");
        LCDWriteStringXY(2,1,"your card!");

        if(byte == CARD_FOUND)
```

```
{

        byte = mfrc522_get_card_serial(str);
        if(byte == CARD_FOUND)
        {
                check=EEpromReadByte(0);
                check1=EEpromReadByte(17);
                LCDClear();
                for(byte=0;byte<8;byte++)
                {
                LCDHexDumpXY(byte*2,0,str[byte]);
                if(str[byte]==check){

                        test=1;
                }
                else if(str[byte]==check1)
                {


                test1=1;
                }}

                if(test==1)
                {
                        DIOSetPinDirection(1,1,1);
                        DIOSetPinValue(1,1,1);
                }
                else if(test1==1){
                        DIOSetPinDirection(1,2,1);

DIOSetPinValue(1,2,1);
                }
                else
                {
                        LCDClear();
                        LCDWriteStringXY(0,0,"Not registered!");
                }


                DIOSetPinDirection(1,0,1);
                DIOSetPinValue(1,0,1);
```

```
                _delay_ms(2500);
                DIOSetPinValue(1,0,0);
                DIOSetPinValue(1,1,0);
                DIOSetPinValue(1,2,0);
            }
            else
            {
                LCDClear();
                LCDWriteStringXY(0,1,"Error");
            }
        }

        _delay_ms(1000);
    }
}
```

- Main program body with application on reading cards and compare it with stored data.

CHAPTER 4

DATABASE

Database is a collection of related data; Data are known facts that can be recorded and have implicit meaning. Is built with a specific purpose (objective) in mind. Database has an intended group of users and some preconceived applications in which these users are interested. A database can be of any size and of varying complexity.

1. Database Management System

The data in the database system is actually stored in the database system, when stored; it must be manipulated using some specific programs.

This collection of programs that enables users to create and maintain a database system is called Database Management System DBMS.

2. Basic functions of DBMS

- Querying the database.
  - Extracting the information that already stored.
- Updating the database.
  - Inserting new customer or details of a new flight. This is controlled by DBMS.
- Generating reports from the data.

3. Database System Environment

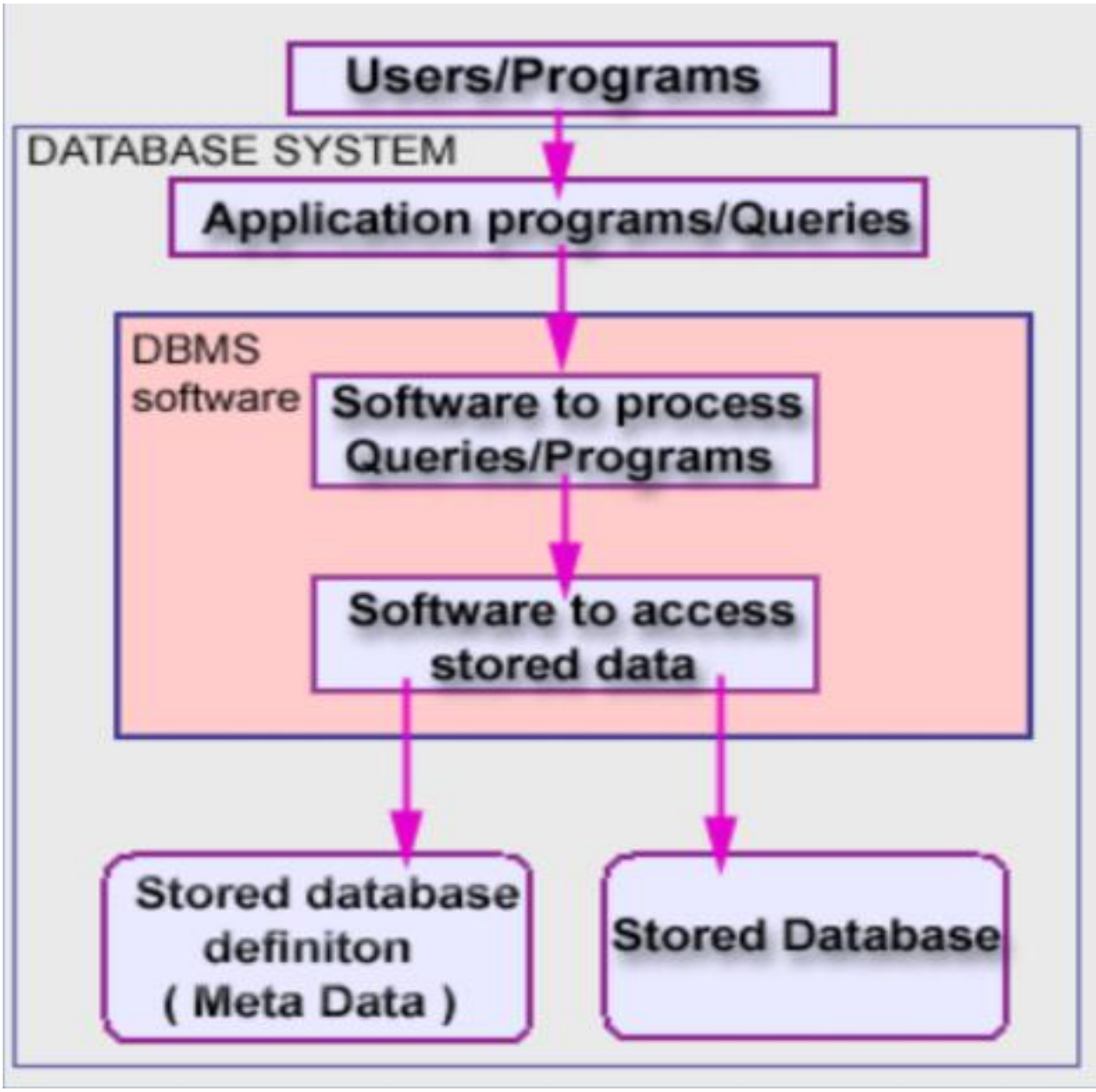


Figure 22: Database System Environment

The structure of data files is stored in the DB Catalogue separately from access programs.

Changing the structure of the files doesn't affect the DBMS software.

4. DB Approaches Characteristics

4.1 Data Abstraction

A data model is used to hide storage details and present the users with a conceptual view of the database.

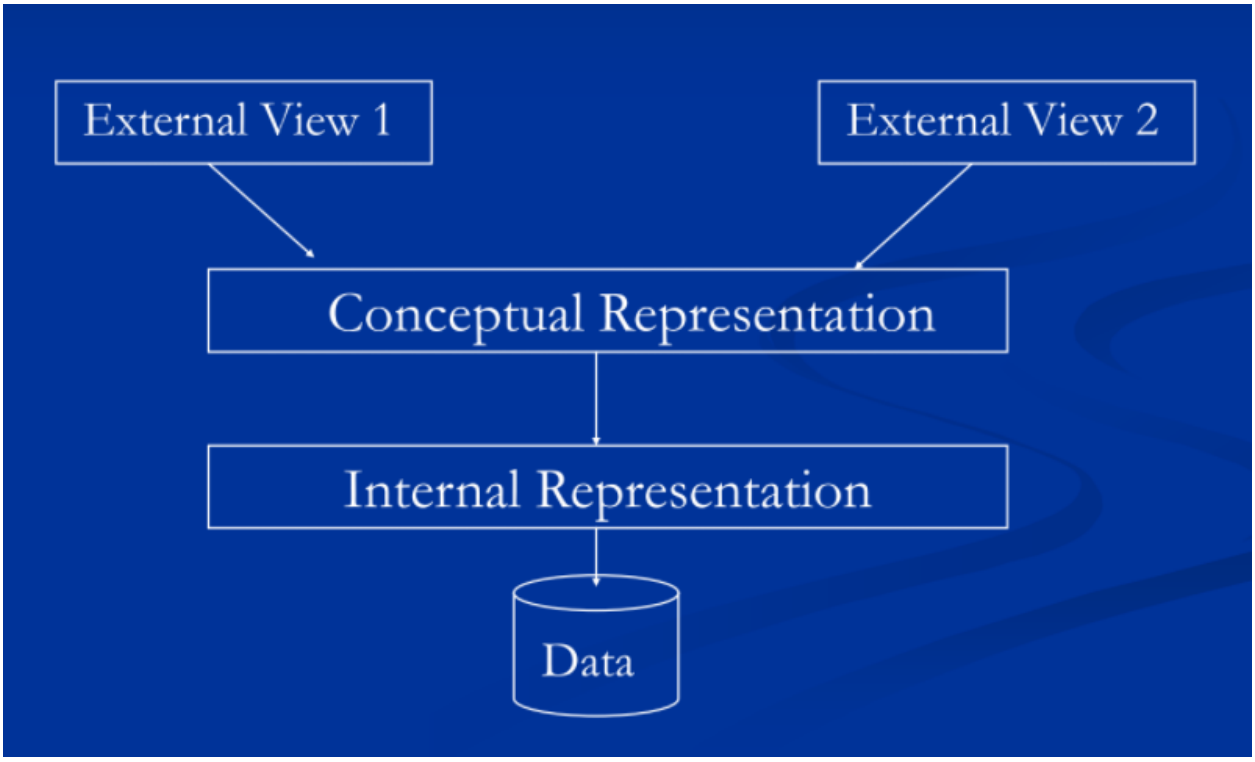


Figure 23: Data Abstraction

4.2 Support of multiple views of the Data

A database has many users, each of whom may require a different perspective or view of the database.

4.3 Multiuser Processing

Multiple users can access the database at the same time.

5. DB Actors

5.1 DB Administrators

- Authorize access to the database and coordinate its use.
- Accountable for security and efficiency problems.
- Acquire software and hardware resources as needed.

5.2 DB Designers

- Identify the data to be stored in the database and choose appropriate structures to represent and store these data.
- Develop views of the database that meets the data and processing requirements of different users.

5.3 End Users

- People whose jobs require access to the database.

5.4 System Analysts and Application Programmers

- Develop and implement specifications based on end user requirements.

5.5 DBMS designers and implementers

5.6 Tool developers

5.7 Operators and maintenance personnel

6. General Description of the Relational Model

- The model design is based on using the mathematical set theory and the first order predicate logic (deductive language).
- It uses a set of relations to describe a database.
- Each relation is viewed as a table of values.
- A tuple corresponds to a specific row (record) in the relation or the table.
- It describes a real-world entity or a relationship.
- It corresponds to the header of a column in a relation.

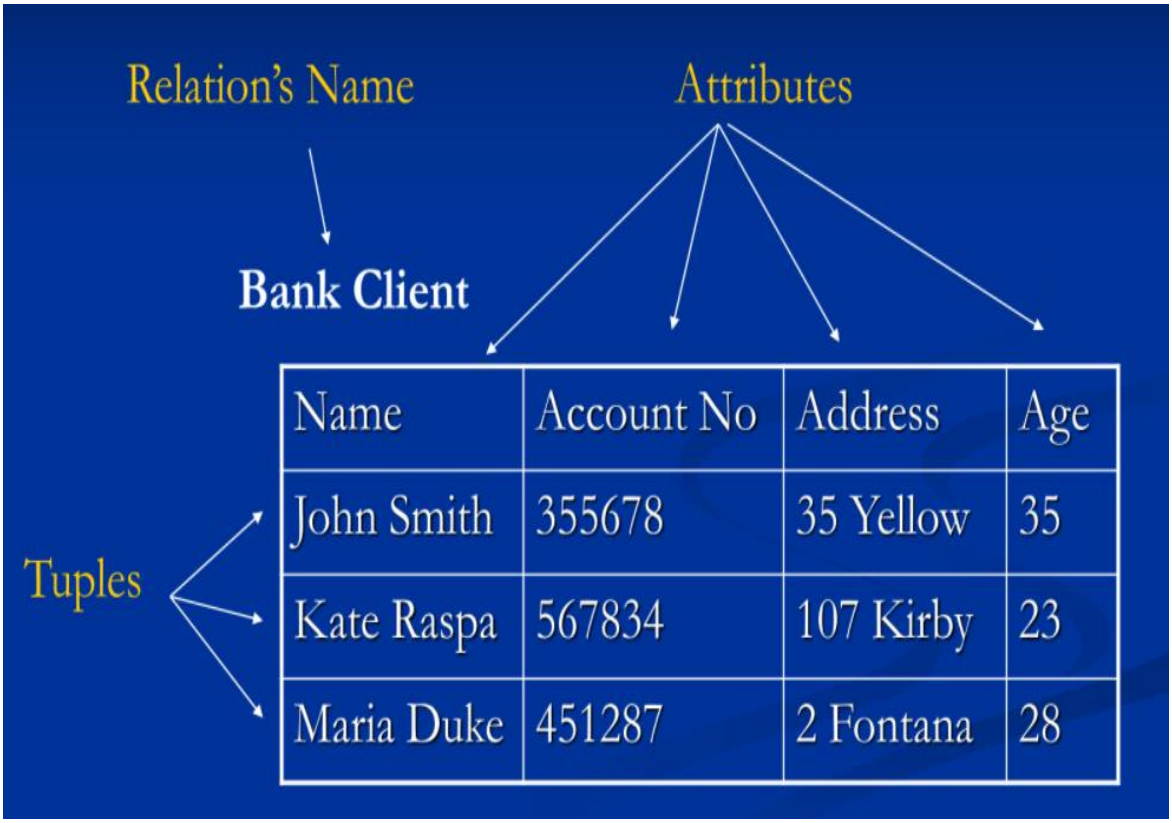


Figure 24: relation model

- Relation (r) is a set of tuples.
- The set of all tuples belonging to a relation (r) is also called relation instance.
- Relation Schema is a table, and attribute is field (Column).
- Relational database schema is a collection of relation schemas.
- Key is the attribute that uniquely identifies a tuple. Each relation may have more than one key, each of which is a Candidate key.
- Super-Key: is a subset of attributes of a relational schema for which in certain instance r there are no two different tuples with the same attribute values. No two distinct tuples in a relation schema R have the same value for the Super Key.
- The value of a primary key must not be null for any tuple that participates in a relation schema.
- A tuple in a relation schema R1 that refers to a relation schema R2 must refer to an existing tuple of R2.

7. SQL

SQL is Schema Definition, Constraints, and Queries and Views. It is used to ask something from a database.

A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

SELECT <attribute list>

- Is a list of attribute names whose values are to be retrieved by the query.

FROM <table list>

- Is a list of the relation names required to process the query.

[WHERE <condition>]

- Is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

[GROUP BY <grouping attribute(s)>]

[HAVING <group condition>]

[ORDER BY <attribute list>]

8. Data types

Characters, Numbers, Times and dates, and Numeric.

9. Basic search commands

- SELECT A1, A2, ..., An (Rows)
- FROM R1, R2, ..., Rn (Tables)
- WHERE C (Conditions)
- * Refers to all fields.

10. Creation, Deletion, and modification of table

- CREATE -> Create a table
    - Description of data types, constraints (if any), field definition, null values etc.
- ALTER -> Alters, is used to modify a table.
- INSERT TABLE inserts one or more rows. We define the name of the relation (table), The list of values (for each column) for the tuple, and Attention to the order of the values.
- DELETE TABLE deletes one/more row/s (tuple/s).
- UPDATE TABLE modifies fields one/more tuple/s.

SQL has 3 major components:

- Data Definition Language (DDL):
  - Definition of the DB structure.
- Data Modeling Language (DML):
  - Retrieving, updating, querying, manipulating data.
- The Data Control Language (DCL).
  - The DCL part of SQL is concerned with Controlling the structure and Accessibility of the database

SQL syntax in general

SQL statements have 2 types of words:

- Reserved words: This is the fixed part of SQL and must be written exactly as required.
- User-defined words: made up by the user and usually are names of database objects.

SQL: General syntax

- SELECT Retrieves data from tables
- INSERT Adds row(s) to a table
- UPDATE Changes field(s) in the records
- DELETE Removes row(s) from a table Data Definition
- CREATE TABLE defines a table and its columns (fields)
- DROP TABLE deletes a table
- ALTER TABLE Adds a new column, add/drop primary key

- CREATE INDEX Creates an index
- DROP INDEX Deletes an index
- CREATE VIEW defines a logical table from other table(s) or view(s)
- DROP VIEW deletes a view

SQL commands – Clauses

Clauses are the commands issued during a query and execute an action in databases and begin each SQL query.

E.g., SELECT, INSERT, ADD, DROP, CREATE etc.

SQL commands – Functions

Functions built into SQL that perform several tasks (add or summing column values, average column values).

SQL commands – Operators

Manipulate numbers and strings or test for equality. 4 kinds (Arithmetic, Range, Equality, Logical).

SQL versus Relational Model

SQL is based on RM.

Main difference:

- SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values.
- SQL relation (table) is a multi-set (sometimes called a bag).

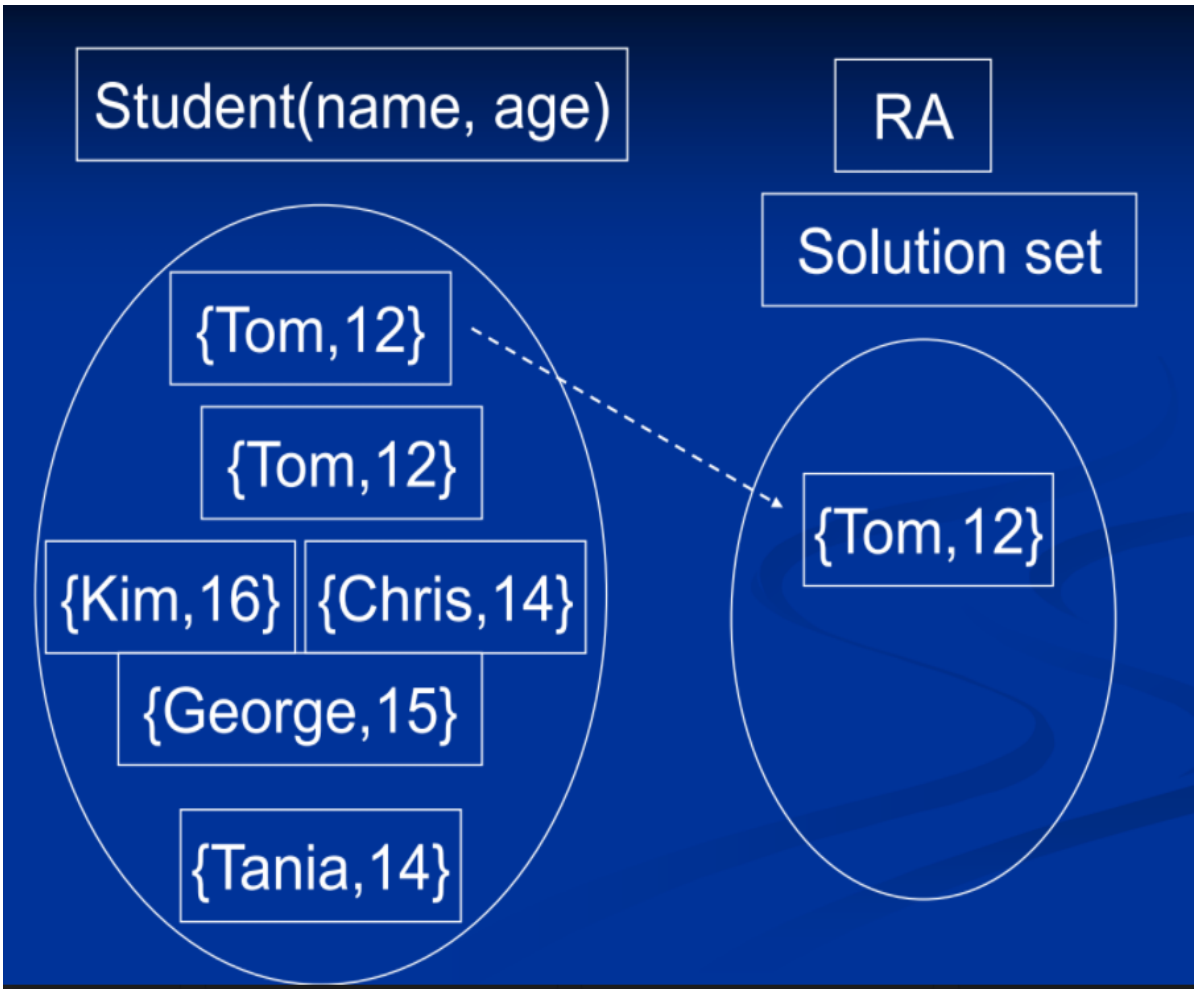


Figure 25: SQL with two identical tuples

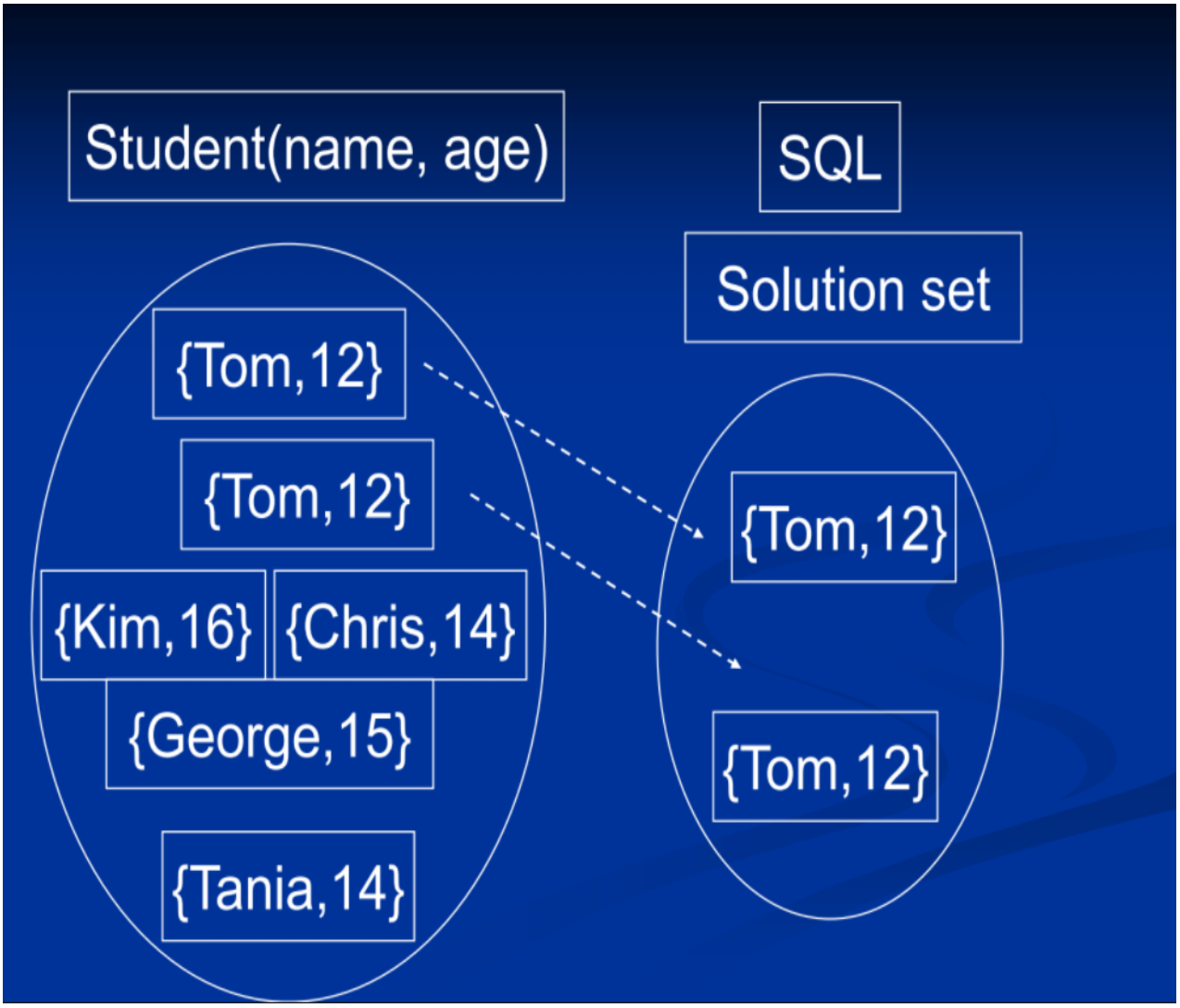


Figure26: multiset SQL

CHAPTER 5

THE MOBILE APPLICATION

The second part of our system is the mobile application. The mobile application is considered the second payment method, that will be through a QR code. Also, many users will be able to know the schedules of buses and how many empty chairs available and other information. This method helps reducing overcrowding in bus stations and so decreasing the spreading of Corona Virus.

1. Advantages

We decided to add an application to our system for several reasons, including:

- Available where most of us have smart phone.
- Easy to use.
- Low cost

2. Objectives

How will our Mobile Application achieve system goals?

1. Decreasing the spreading of Corona Virus.

The use and exchange of paper money between people makes it one of the reasons for the spread of the virus as well. So, we changed the payment method to done by scanning a QR code using the mobile application. Each

user pays via his mobile phone by QR Code that contains the bus ID, the chair number and the price of the trip.

2. Reducing overcrowding in bus stations.

The application shows buses time by using GPS, considering that the bus is a moving point, and the station is a fixed point, and we have the average speed of the bus, and from here we can calculate the estimated time for the bus to arrive at the station and so the number of people at the station can be reduced as it became known when the bus will arrive and there is no need to be at the station before or after this time

The application also shows the number of the available chairs. When each user makes a scan, either with the smart card or with the mobile application, this information is stored in the database, and from here the number of empty chairs is identified. And so, there is also no need to be at the station if there are no available chairs at the bus.

In our mobile application we used **Flutter** to implement it.

3. What is flutter?

Flutter is a free and open-source mobile UI framework created by Google and released in May 2017. In a few words, it allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create an app that is suitable for the two-operating system iOS and Android both.

-Flutter consists of two important parts:

- **An SDK (Software Development Kit):** A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
- **A Framework (UI Library based on widgets):** A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

4. Dart

To develop with Flutter, you will use a programming language called **Dart**. The language was created by Google in October 2011, but it has improved a lot over these past years.

5. Why we used flutter?

1- Cross Platform

Flutter is a developmental tool that is compatible across multiple platforms. It is resource-efficient and preferred by developers. Software developers can utilize the same code base for creating Android and iOS applications. Cross-platform development helps to reduce resource utilization and saves a lot of time.

2- Simple to learn and use

Flutter is a modern framework. It is way simpler to create mobile applications with same efficiency of (Kotlin, Java) in Android and (swift, objective c) in IOS.

3- Quick compilation: maximum productivity

you can change your code and see the results in real-time. It is called Hot-Reload. It only takes a short amount of time after you save to update the application itself. Significant modifications force you to reload the app. But if you do work like design, for example, and change the size of an element, it is in real-time!

5. Application design

The Mobile Application that will be with the users has many functions such as log-in/sign-up function, searching or locating function, payment or scanning QR code function and other functions.

Figure27: logo of the app

- Log-in:

If the user has registered before and has an account on the application, then he will be asked to enter his email and password then pressing on log-in button.

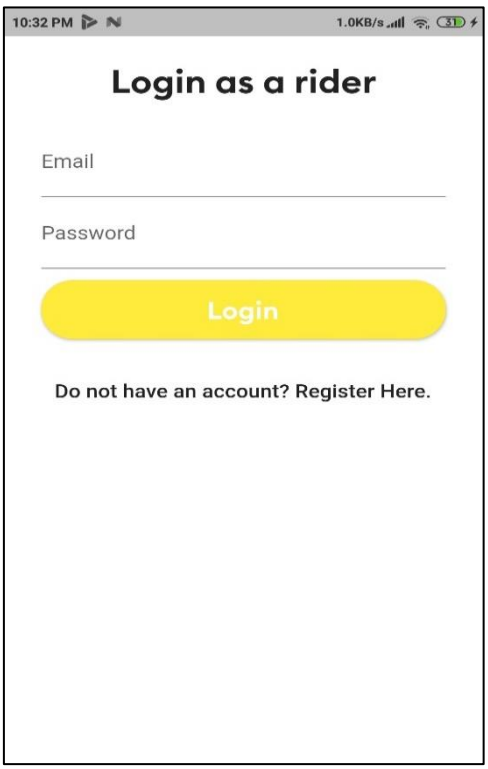


Figure28: log-in screen

- Sign-up:

If the user has not an account, he can press on “register here”. He will fill the required data (Name, email, phone number and password) then he presses on “create account” button and he will have a new account.

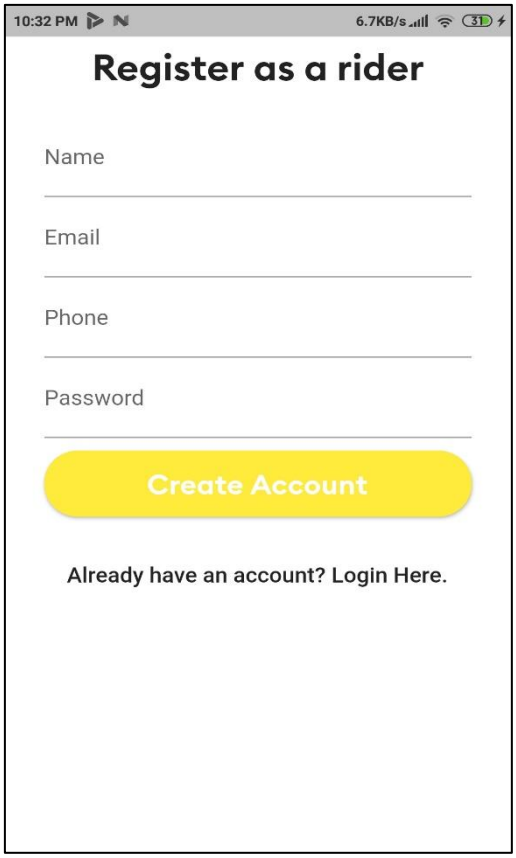


Figure29: registration screen

- Home page

At the main page of the app there will appear a search bar and map that the user can use to search for the places that he wants.

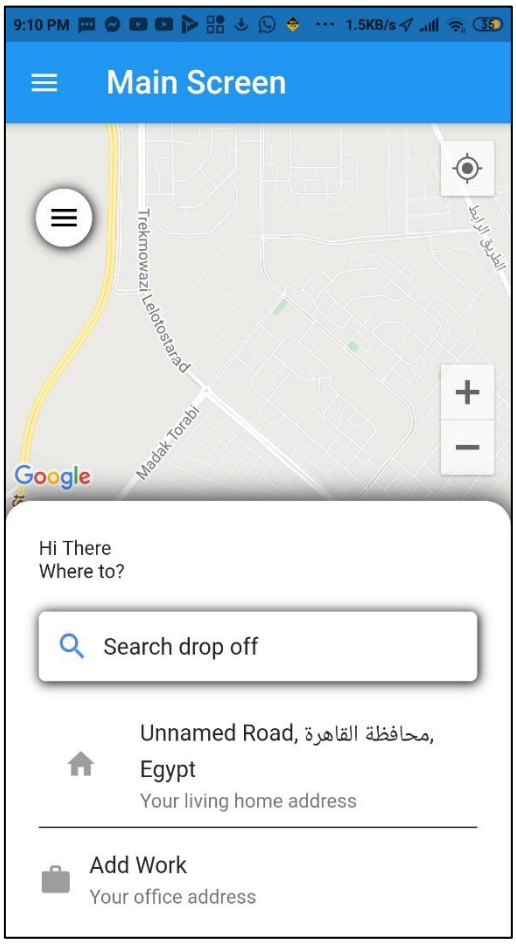


Figure 30: app main screen

- Search screen

When the user enters the location and destination the app will show him all the buses and different routes to his destination and all the information needed to know of the selected bus. This information such as meeting point, bus leaving time, estimated arrival time, ticket price and no. of available seats in the bus. And this is the first use of the mobile application.

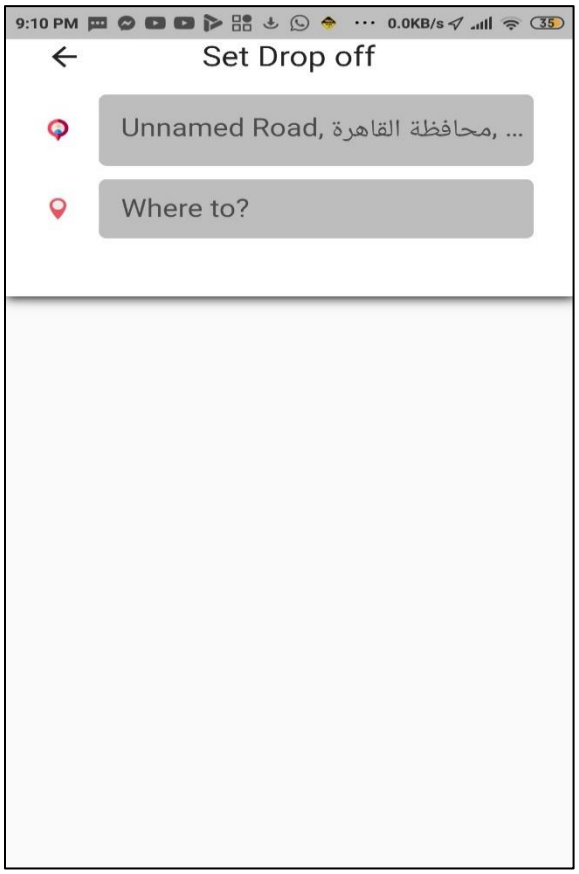


Figure31: search screen

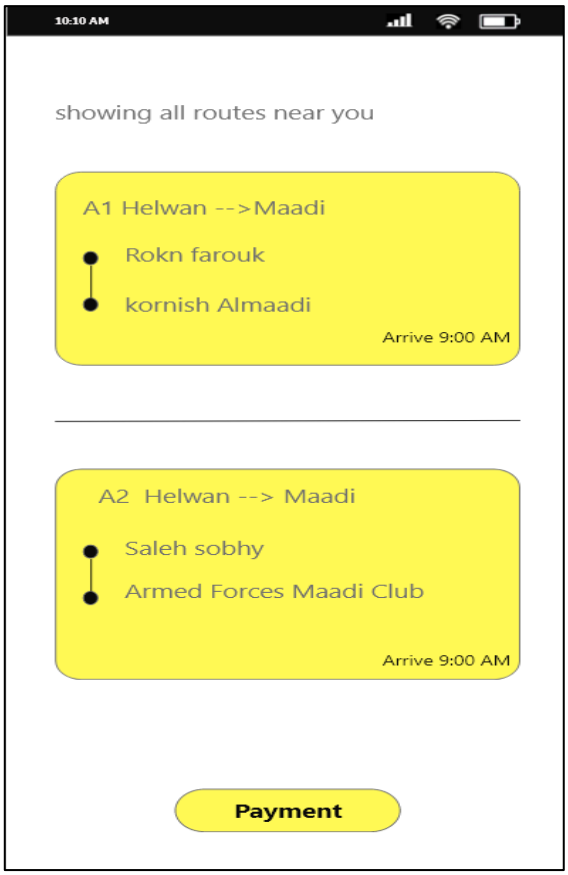


Figure32: buses routes

- payment

There is also a side menu which contains some options that helps the user. One of them is the payment option which is the second way to pay the bus ticket in our system. When the user presses on payment the app will ask him to enter the required information of the visa or the card that will relate to the app and from it all the dealings will be done.

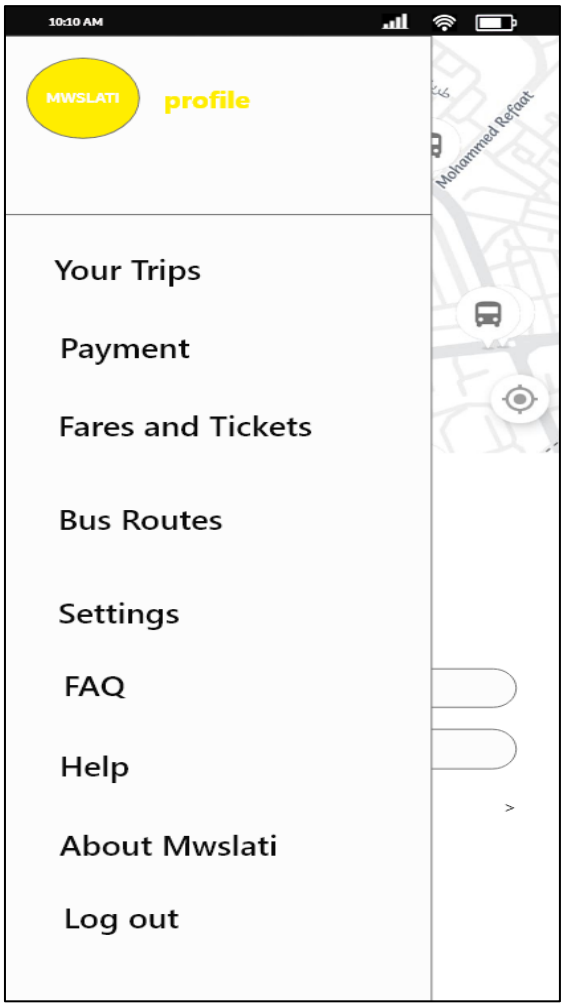


Figure33: side menu

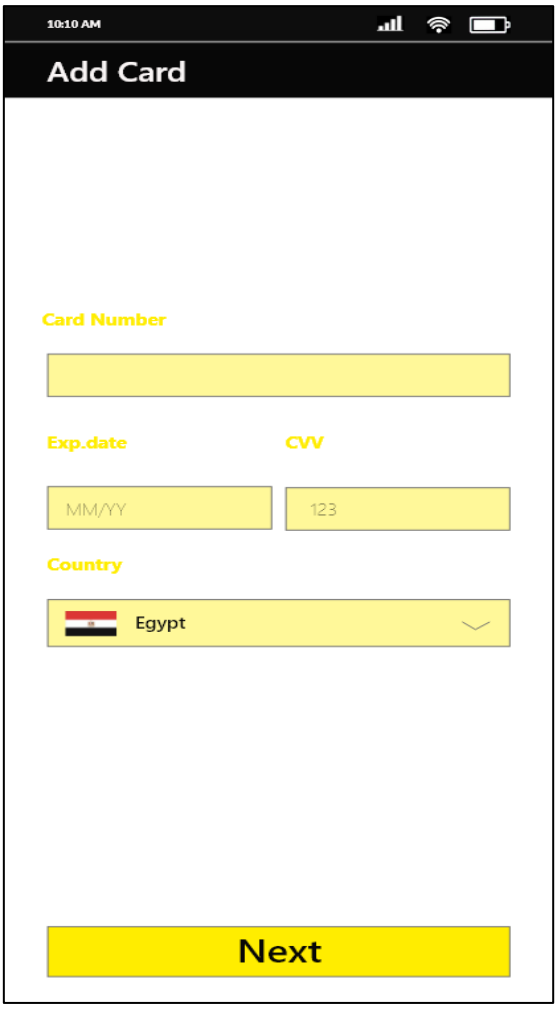


Figure34: card information

- QR code scanning

After the user enters this information, he will be moved to the QR code scanning page. At where the user can pay the bus ticket by scanning the QR code that is on each chair at the bus. After the scanning, the lock of the chair will be opened, and ticket will be paid, and the user can use the chair.

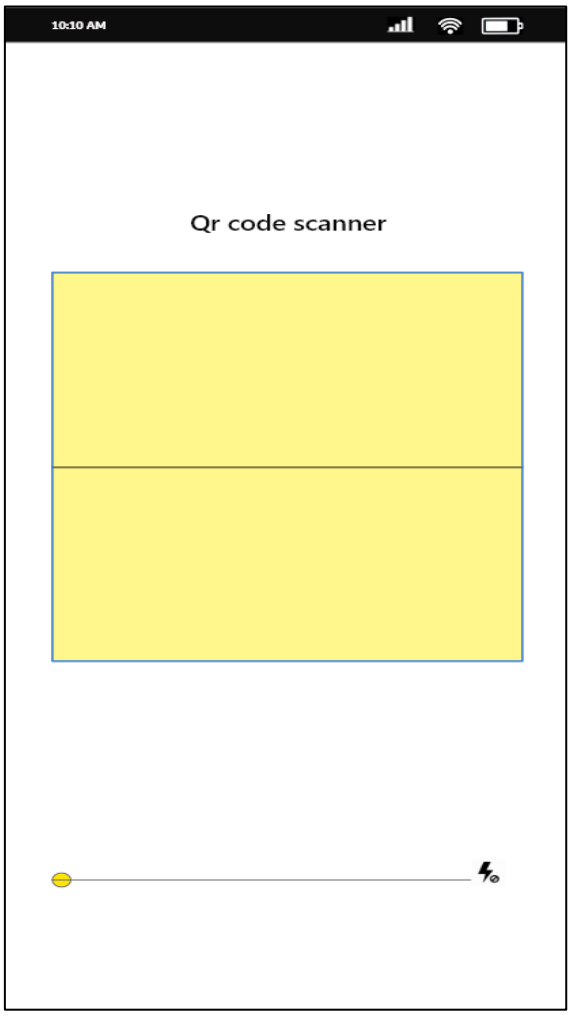


Figure 35: QR code scanner

CHAPTER 6

PROJECT COST ANALYSIS

The project cost is divided into two types of cost

1. Fixed cost

It is the cost of things that will be used one time for the whole project. It is like the code developing cost, this code will be applied to all the project devices that will be used no matter how many it is.

2. Variable Cost

It is the cost of hardware components that are used in each bus in our case and the cost of renting a web server and the network carrier cost. The hardware cost:

- Controller Atmega 32 = 100EGP.
- GSM module sim900a = 500EGP.
- GPS module Neo-6m =500EGP.
- Wi-fi module esp8266 = 300 EGP.
- RFID module mfrc522 = 400 EGP.
- PCB design = 200EGP.
- Containing box = 150EGP.
- Lm2596 buck converter = 50 EGP
- LCD 16*2 = 40EGP.
- EEPROM = 50EGP.

This is the cost of the hardware parts used in this project,

We will discuss the priority of each part.

PART	PRIORITY
CONTROLLER ATMEGA 32	High
RFID	High
LCD	High
GSM MODULE	Medium(limited functions)
GPS	Medium(limited functions)
WIFI MODULE	Medium(limited functions)
EEPROM	High (data storage)
LM2596 BUCK CONVERTER	High(power device)

Table 1: The priority of each part in the hardware

CHAPTER 7

APPLICATION CODE IMPLEMENTATION

1. The code of log-in screen

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:mwslati3/AllWidgets/progressDialog.dart';
import 'package:mwslati3/Allscreens/mainscreen.dart';
import 'package:mwslati3/Allscreens/registerationScreen.dart';
import 'package:mwslati3/main.dart';
import 'package:mwslati3/sharedPreferences.dart';

class LoginScreen extends StatelessWidget {
  static const String idScreen = "Login";
  TextEditingController emailTextEditingContoller = TextEditingController();
  TextEditingController passwordTextEditingContoller = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      body: SingleChildScrollView(
        child: Padding(
          padding: EdgeInsets.all(8.0),
          child: Column(
            children: [
              SizedBox(
                height: 35.0,
              ),
              Image(
                image: AssetImage("images/logo.png"),
                width: 390.0,
                height: 258.0,
                alignment: Alignment.center,
              ),

              SizedBox(
                height: 1.0,
              ),
```

```
37                  Text(
38                    "Login as a rider",
39                    style: TextStyle(fontSize: 24.0, fontFamily: "Brand Bold"),
40                    textAlign: TextAlign.center,
41                  ),
42                  Padding(
43                    padding: EdgeInsets.all(20.0),
44                    child: Column(
45                      children: [
46                        SizedBox(
47                          height: 1.0,
48                        ),
49                        TextField(
50                          controller: emailTextEditingContoller,
51                          keyboardType: TextInputType.emailAddress,
52                          decoration: InputDecoration(
53                            labelText: "Email",
54                            labelStyle: TextStyle(
55                              fontSize: 14.0,
56                            ),
57                            hintStyle: TextStyle(
58                              color: Colors.grey,
59                              fontSize: 10.0,
60                            ),
61                          ),
62                          style: TextStyle(fontSize: 14.0),
63                        ),
64                        SizedBox(
65                          height: 1.0,
66                        ),
67                        TextField(
68                          controller: passwordTextEditingContoller,
69                          obscureText: true,
70                          decoration: InputDecoration(
71                            labelText: "Password",
72                            labelStyle: TextStyle(
73                              fontSize: 14.0,
74                            ),
75                            hintStyle: TextStyle(
76                              color: Colors.grey,
77                              fontSize: 10.0,
78                            ),
79                          ),
80                          style: TextStyle(fontSize: 14.0),
81                        ),
82                        SizedBox(
83                          height: 10.0,
84                        ),
85                        // ignore: deprecated_member_use
86                        RaisedButton(
87                          color: Colors.yellow,
88                          textColor: Colors.white,
89                          child: Container(
90                            height: 50.0,
91                            child: Center(
92                              child: Text(
93                                "Login",
94                                style: TextStyle(
95                                  fontSize: 18.0, fontFamily: "Brand Bold"),
96                              ),
97                            ),
98                          ),
99                          shape: new RoundedRectangleBorder(
100                           borderRadius: new BorderRadius.circular(24.0),
101                         ),
102                         onPressed: () {
103                           if (!emailTextEditingContoller.text.contains("@")) {
104                             displayToastMessage(
105                               "Email address is not valid.", context);
```

```
106                     } else if (passwordTextEditingContoller.text.isEmpty) {
107                       displayToastMessage(
108                           "password is mandatory.", context);
109                     } else {
110                       loginAndAuthenticationUser(context);
111                     }
112                   },
113                 ),
114               ],
115             ),
116           ),
117           // ignore: deprecated_member_use
118           FlatButton(
119             onPressed: () {
120               Navigator.pushNamedAndRemoveUntil(
121                   context, RegisterationScreen.idScreen, (route) => false);
122             },
123             child: Text(
124               "Do not have an account? Register Here.",
125             ),
126           ),
127         ],
128       ),
129     ),
130   ),
131 );
132 }
133
134 final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
135
136 void loginAndAuthenticationUser(BuildContext context) async {…
177 }
178 }
```

2. The code of registration screen

```
1   import 'package:firebase_auth/firebase_auth.dart';
2   import 'package:firebase_core/firebase_core.dart';
3   import 'package:flutter/material.dart';
4   import 'package:fluttertoast/fluttertoast.dart';
5   import 'package:mwslati3/AllWidgets/progressDialog.dart';
6   import 'package:mwslati3/Allscreens/loginScreen.dart';
7   import 'package:mwslati3/Allscreens/mainscreen.dart';
8   import 'package:mwslati3/main.dart';
9
10  import '../sharedPreferences.dart';
11
12  class RegisterationScreen extends StatelessWidget {
13    static const String idScreen = "Register";
14    TextEditingController nameTextEditingContoller = TextEditingController();
15    TextEditingController emailTextEditingContoller = TextEditingController();
16    TextEditingController phoneTextEditingContoller = TextEditingController();
17    TextEditingController passwordTextEditingContoller = TextEditingController();
18
19    @override
20    Widget build(BuildContext context) {
21      return Scaffold(
22        backgroundColor: Colors.white,
23        body: SingleChildScrollView(
24          child: Padding(
25            padding: EdgeInsets.all(8.0),
26            child: Column(
27              children: [
28                SizedBox(
29                  height: 20.0,
30                ),
31                Image(
32                  image: AssetImage("images/logo.png"),
33                  width: 390.0,
34                  height: 258.0,
35                  alignment: Alignment.center,
36                ),
37                SizedBox(
38                  height: 1.0,
39                ),
40                Text(
41                  "Register as a rider",
42                  style: TextStyle(fontSize: 24.0, fontFamily: "Brand Bold"),
43                  textAlign: TextAlign.center,
44                ),
45                Padding(
46                  padding: EdgeInsets.all(20.0),
47                  child: Column(
48                    children: [
49                      SizedBox(
50                        height: 1.0,
51                      ),
52                      TextField(
53                        controller: nameTextEditingContoller,
54                        keyboardType: TextInputType.text,
55                        decoration: InputDecoration(
56                          labelText: "Name",
57                          labelStyle: TextStyle(
58                            fontSize: 14.0,
59                          ),
60                          hintStyle: TextStyle(
61                            color: Colors.grey,
62                            fontSize: 10.0,
63                          ),
64                        ),
65                        style: TextStyle(fontSize: 14.0),
66                      ),
67                      SizedBox(
68                        height: 1.0,
69                      ),
70                      TextField(
71                        controller: emailTextEditingContoller,
```

```
keyboardType: TextInputType.emailAddress,
decoration: InputDecoration(
  labelText: "Email",
  labelStyle: TextStyle(
    fontSize: 14.0,
  ),
  hintStyle: TextStyle(
    color: Colors.grey,
    fontSize: 10.0,
  ),
),
style: TextStyle(fontSize: 14.0),
),
SizedBox(
  height: 1.0,
),
TextField(
  controller: phoneTextEditingContoller,
  keyboardType: TextInputType.phone,
  decoration: InputDecoration(
    labelText: "Phone",
    labelStyle: TextStyle(
      fontSize: 14.0,
    ),
    hintStyle: TextStyle(
      color: Colors.grey,
      fontSize: 10.0,
    ),
  ),
  style: TextStyle(fontSize: 14.0),
),
SizedBox(
  height: 1.0,
),
```

```
controller: passwordTextEditingContoller,
obscureText: true,
decoration: InputDecoration(
  labelText: "Password",
  labelStyle: TextStyle(
    fontSize: 14.0,
  ),
  hintStyle: TextStyle(
    color: Colors.grey,
    fontSize: 10.0,
  ),
),
style: TextStyle(fontSize: 14.0),
),
SizedBox(
  height: 10.0,
),
RaisedButton(
  color: Colors.yellow,
  textColor: Colors.white,
  child: Container(
    height: 50.0,
    child: Center(
      child: Text(
        "Create Account",
        style: TextStyle(
          fontSize: 18.0, fontFamily: "Brand Bold"),
      ),
    ),
  ),
  shape: new RoundedRectangleBorder(
    borderRadius: new BorderRadius.circular(24.0),
  ),
  onPressed: () {
```

```
141                     if (nameTextEditingContoller.text.length < 3) {
142                       displayToastMessage(
143                           "name must be atleast 3 characters.", context);
144                     } else if (!emailTextEditingContoller.text
145                         .contains("@")) {
146                       displayToastMessage(
147                           "Email address is not valid.", context);
148                     } else if (phoneTextEditingContoller.text.isEmpty) {
149                       displayToastMessage(
150                           "phone Number is mandatory.", context);
151                     } else if (passwordTextEditingContoller.text.length <
152                         6) {
153                       displayToastMessage(
154                           "password must be at least 6 characters.",
155                           context);
156                     } else {
157                       registrNewUser(context);
158                     }
159                   },
160                 ),
161               ],
162             ),
163           ),
164           FlatButton(
165             onPressed: () {
166               Navigator.pushNamedAndRemoveUntil(
167                   context, LoginScreen.idScreen, (route) => false);
168             },
169             child: Text(
170               "Already have an account? Login Here.",
171             ),
172           ),
173         ],
174       ),
175     ),
176   ),
177 );
178 }
179
180 final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;
181 void registrNewUser(BuildContext context) async {
182   showDialog(
183       context: context,
184       barrierDismissible: false,
185       builder: (BuildContext context) {
186         return progressDialog(
187           message: "Regisreing,please wait...",
188         );
189       });
190   final User? firebaseUser = (await _firebaseAuth
191           .createUserWithEmailAndPassword(
192               email: emailTextEditingContoller.text,
193               password: passwordTextEditingContoller.text)
194           .catchError((errMsg) {
195     Navigator.pop(context);
196
197     displayToastMessage("Error:" + errMsg.toString(), context);
198   }))
199       .user;
200   if (firebaseUser != null) {
201     //save user info to data base
202     Map userDataMap = {
203       "name": nameTextEditingContoller.text.trim(),
204       "email": emailTextEditingContoller.text.trim(),
205       "phone": phoneTextEditingContoller.text.trim(),
206     };
207     usersRef.child(firebaseUser.uid).set(userDataMap);
```

```
208         displayToastMessage("your account has been created.", context);
209         MySharedPreferences.saveUserSingIn(true);
210
211         Navigator.pushNamedAndRemoveUntil(
212             context, MainScreen.idScreen, (route) => false);
213       } else {
214         Navigator.pop(context);
215
216         //error occured-display error msg
217         displayToastMessage("New user account has not been created.", context);
218       }
219     }
220   }
221
222   displayToastMessage(String message, BuildContext context) {
223     Fluttertoast.showToast(msg: message);
224   }
```

3. The code of main screen

```
1   import 'dart:async';
2   import 'package:auto_size_text/auto_size_text.dart';
3   import 'package:firebase_auth/firebase_auth.dart';
4   import 'package:flutter/cupertino.dart';
5   import 'package:geolocator/geolocator.dart';
6   import 'package:flutter/material.dart';
7   import 'package:google_maps_flutter/google_maps_flutter.dart';
8   import 'package:mwslati3/AllWidgets/Divider.dart';
9   import 'package:mwslati3/Allscreens/searchScreen.dart';
10  import 'package:mwslati3/Assistants/assistantMethod.dart';
11  import 'package:mwslati3/DataHandler/appData.dart';
12  import 'package:mwslati3/Models/users.dart';
13
14  import 'package:mwslati3/sharedPreferences.dart';
15  import 'package:mwslati3/splashscreen.dart';
16  import 'package:provider/provider.dart';
17
18  class MainScreen extends StatefulWidget {
19    static const String idScreen = "mainScreen";
20
21    _MainScreenState createState() => _MainScreenState();
22  }
23
24  class _MainScreenState extends State<MainScreen> {
25    Completer<GoogleMapController> _controllerGoogleMap = Completer();
26    late GoogleMapController newGoogleMapController;
27    GlobalKey<ScaffoldState> scaffoldKey = new GlobalKey<ScaffoldState>();
28    late Position currentPosition;
29    var geolocator = Geolocator();
30    double bottomPaddingOfMap = 0;
31    void locatePosition() async {
32      Position position = await Geolocator.getCurrentPosition(
33          desiredAccuracy: LocationAccuracy.high);
34      currentPosition = Position as Position;
35      LatLng latLaPosition = LatLng(position.latitude, position.longitude);
36      CameraPosition cameraPosition =
```

```
37          new CameraPosition(target: latLaPosition, zoom: 14);
38      newGoogleMapController
39          .animateCamera(CameraUpdate.newCameraPosition(cameraPosition));
40      String address =
41          await AssistantMethods.searchCoordinateAddress(position, context);
42      print("this is your Address ::" + address);
43    }
44
45    static final CameraPosition _kGooglePlex = CameraPosition(
46      target:
47          LatLng((double.parse(Users.userLat!)), (double.parse(Users.userLong!))),
48      zoom: 14.4746,
49    );
50    static final CameraPosition _kGooglePlex2 = CameraPosition(
51      target:
52          LatLng((double.parse(Users.userLat!)), (double.parse(Users.userLong!))),
53      zoom: 14.4746,
54    );
55
56    getUserData() async {
57      Users.userlogIn = await MySharedPreferences.getUserSingIn() ?? false;
58      Users.userAdders = await MySharedPreferences.getUserAddress();
59      Users.userLat = await MySharedPreferences.getUserLat();
60      Users.userLong = await MySharedPreferences.getUserlong();
61    }
62
63    @override
64    void initState() {
65      super.initState();
66
67      getUserData();
68    }
69
70    Widget build(BuildContext context) {
71      print(Users.userLat);
72        return Scaffold(
73          key: scaffoldKey,
74          appBar: AppBar(
75            title: Text("Main Screen"),
76          ),
77          drawer: Container(
78            color: Colors.white,
79            width: 255.0,
80            child: Drawer(
81              child: ListView(
82                children: [
83                  //Drawer Header
84                  Container(
85                    height: 165.0,
86                    child: DrawerHeader(
87                      decoration: BoxDecoration(color: Colors.white),
88                      child: Row(
89                        children: [
90                          Image.asset(
91                            "images/user_icon.png",
92                            height: 65.0,
93                            width: 65.0,
94                          ),
95                          SizedBox(
96                            width: 16.0,
97                          ),
98                          Column(
99                            mainAxisAlignment: MainAxisAlignment.center,
100                           children: [
101                             Text(
102                               "profile Name",
103                               style: TextStyle(
104                                   fontSize: 16.0, fontFamily: "Brand-Bold"),
105                             ),
```

```
106                                 SizedBox(
107                                   height: 6.0,
108                                 ),
109                                 Text("Visit profile"),
110                               ],
111                             ),
112                           ],
113                         ),
114                       ),
115                     ),
116                     DividerWidget(),
117                     SizedBox(
118                       height: 12.0,
119                     ),
120                     //Drawer Body Controllers
121                     ListTile(
122                       leading: Icon(Icons.history),
123                       title: Text(
124                         "History",
125                         style: TextStyle(fontSize: 15.0),
126                       ),
127                     ),
128                     ListTile(
129                       leading: Icon(Icons.person),
130                       title: Text(
131                         "Visit profile",
132                         style: TextStyle(fontSize: 15.0),
133                       ),
134                     ),
135                     ListTile(
136                       leading: Icon(Icons.info),
137                       title: Text(
138                         "About",
139                         style: TextStyle(fontSize: 15.0),
140                       ),
141                     ),
142                   ],
143                 ),
144               ),
145             ),
146             body: (Users.userLat == null)
147                 ? Center(
148                     child: Container(
149                       child: MaterialButton(
150                         onPressed: () async {
151                           try {
152                             final geoposition = await Geolocator.getCurrentPosition(
153                               desiredAccuracy: LocationAccuracy.high,
154                             );
155                             setState(() {
156                               MySharedPreferences.saveUserlong(
157                                   '${geoposition.longitude}');
158                               MySharedPreferences.saveUserlat(
159                                   '${geoposition.latitude}');
160                             });
161                             Navigator.of(context)
162                                 .pushReplacementNamed(SplashScreen.route);
163                           } catch (e) {
164                             print('geoposition Erorr:' + e.toString());
165                           }
166                         },
167                         color: Colors.blueAccent,
168                         child: Text(
169                           'Please Allow your location first..',
170                           style: TextStyle(
171                             color: Colors.white,
172                             fontSize: 15,
173                           ),
174                         ),
```

```
175                 ),
176               ),
177             )
178           : Stack(
179               children: [
180                 GoogleMap(
181                   padding: EdgeInsets.only(bottom: bottomPaddingOfMap),
182                   mapType: MapType.normal,
183                   myLocationButtonEnabled: true,
184                   initialCameraPosition: _kGooglePlex,
185                   myLocationEnabled: true,
186                   zoomGesturesEnabled: true,
187                   zoomControlsEnabled: true,
188                   onMapCreated: (GoogleMapController controller) {
189                     _controllerGoogleMap.complete(controller);
190                     newGoogleMapController = controller;
191
192                     setState(() {
193                       bottomPaddingOfMap = 300.0;
194                     });
195
196                     locatePosition();
197                   },
198                 ),
199                 //HamburgerButton for Drawer
200                 Positioned(
201                   top: 45.0,
202                   left: 22.0,
203                   child: GestureDetector(
204                     onTap: () {
205                       scaffoldKey.currentState!.openDrawer();
206                     },
207                     child: Container(
208                       decoration: BoxDecoration(
209                         color: Colors.white,
210                         borderRadius: BorderRadius.circular(22.0),
211                         boxShadow: [
212                           BoxShadow(
213                             color: Colors.black,
214                             blurRadius: 6.0,
215                             spreadRadius: 0.5,
216                             offset: Offset(
217                               0.7,
218                               0.7,
219                             ),
220                           ),
221                         ],
222                       ),
223                       child: CircleAvatar(
224                         backgroundColor: Colors.white,
225                         child: Icon(
226                           Icons.menu,
227                           color: Colors.black,
228                         ),
229                         radius: 20.0,
230                       ),
231                     ),
232                   ),
233                 ),
234                 Positioned(
235                   left: 0.0,
236                   right: 0.0,
237                   bottom: 0.0,
238                   child: Container(
239                     height: 300.0,
240                     decoration: BoxDecoration(
241                       color: Colors.white,
242                       borderRadius: BorderRadius.only(
243                           topLeft: Radius.circular(18.0),
244                           topRight: Radius.circular(18.0)),
245                       boxShadow: [
```

```
246                 BoxShadow(
247                   color: Colors.black,
248                   blurRadius: 16.0,
249                   spreadRadius: 0.5,
250                   offset: Offset(0.7, 0.7),
251                 ),
252               ],
253             ),
254             child: Padding(
255               padding: const EdgeInsets.symmetric(
256                   horizontal: 24.0, vertical: 18.0),
257               child: Column(
258                 crossAxisAlignment: CrossAxisAlignment.start,
259                 children: [
260                   SizedBox(
261                     height: 6.0,
262                   ),
263                   Text(
264                     "Hi There",
265                     style: TextStyle(fontSize: 12.0),
266                   ),
267                   Text(
268                     "Where to?",
269                     style: TextStyle(fontSize: 12.0),
270                   ),
271                   SizedBox(
272                     height: 20.0,
273                   ),
274                   GestureDetector(
275                     onTap: () {
276                       Navigator.push(
277                           context,
278                           MaterialPageRoute(
279                               builder: (context) => SearchScreen()));
280                     },
281                     child: Container(
282                       decoration: BoxDecoration(
283                         color: Colors.white,
284                         borderRadius: BorderRadius.circular(5.0),
285                         boxShadow: [
286                           BoxShadow(
287                             color: Colors.black,
288                             blurRadius: 6.0,
289                             spreadRadius: 0.5,
290                             offset: Offset(0.7, 0.7),
291                           ),
292                         ],
293                       ),
294                       child: Padding(
295                         padding: const EdgeInsets.all(12.0),
296                         child: Row(
297                           children: [
298                             Icon(
299                               Icons.search,
300                               color: Colors.blueAccent,
301                             ),
302                             SizedBox(
303                               width: 10.0,
304                             ),
305                             Text("Search drop off")
306                           ],
307                         ),
308                       ),
309                     ),
310                   ),
311                   SizedBox(height: 24.0),
312                   Row(
313                     children: [
314                       Expanded(
```

```
                                  child: Icon(
                                    Icons.home,
                                    color: Colors.grey,
                                  ),
                                ),
                                SizedBox(
                                  width: 12.0,
                                ),
                                Expanded(
                                  flex: 4,
                                  child: Column(
                                    crossAxisAlignment: CrossAxisAlignment.start,
                                    children: [
                                      AutoSizeText(
                                        Provider.of<AppData>(context,
                                                listen: false)
                                            .userAddress
                                            .toString(),
                                      ),
                                      SizedBox(
                                        height: 4.0,
                                      ),
                                      Text(
                                        "Your living home address",
                                        style: TextStyle(
                                            color: Colors.black54,
                                            fontSize: 12.0),
                                      ),
                                    ],
                                  ),
                                )
                              ],
                            ),
                            SizedBox(height: 10.0),
                            DividerWidget(),
```

```
                            SizedBox(
                              height: 16.0,
                            ),
                            Row(
                              children: [
                                Icon(
                                  Icons.work,
                                  color: Colors.grey,
                                ),
                                SizedBox(
                                  width: 12.0,
                                ),
                                Column(
                                  crossAxisAlignment: CrossAxisAlignment.start,
                                  children: [
                                    Text("Add Work"),
                                    SizedBox(
                                      height: 4.0,
                                    ),
                                    Text(
                                      "Your office address",
                                      style: TextStyle(
                                          color: Colors.black54, fontSize: 12.0),
                                    ),
                                  ],
                                )
                              ],
                            ),
                          ],
                        ),
                      ),
                    ),
                  ),
                ],
              ),
      );
```

4. The code of search screen

```
1   import 'package:flutter/material.dart';
2   import 'package:mwslati3/DataHandler/appData.dart';
3   import 'package:provider/provider.dart';
4
5   class SearchScreen extends StatefulWidget {
6     _SearchScreenState createState() => _SearchScreenState();
7   }
8
9   class _SearchScreenState extends State<SearchScreen> {
10    TextEditingController pickUpTextEditingController = TextEditingController();
11    TextEditingController dropOffEditingController = TextEditingController();
12    Widget build(BuildContext context) {
13      // String PlaceAddress =Provider.of<AppData>(context).pickUpLocation.PlaceName??"";
14      // pickUpTextEditingController.text=PlaceAddress;
15
16      return Scaffold(
17        body: Column(
18          children: [
19            Container(
20              height: 215.0,
21              decoration: BoxDecoration(
22                color: Colors.white,
23                boxShadow: [
24                  BoxShadow(
25                    color: Colors.black,
26                    blurRadius: 6.0,
27                    spreadRadius: 0.5,
28                    offset: Offset(0.7, 0.7)),
29                ],
30              ),
31              child: Padding(
32                padding: EdgeInsets.only(
33                    left: 25.0, top: 20.0, right: 25.0, bottom: 20.0),
34                child: Column(
35                  children: [
36                    SizedBox(height: 5.0),
37                    Stack(
38                      children: [
39                        GestureDetector(
40                          onTap: () {
41                            Navigator.pop(context);
42                          },
43                          child: Icon(Icons.arrow_back),
44                        ),
45                        Center(
46                          child: Text(
47                            "Set Drop off",
48                            style: TextStyle(
49                                fontSize: 18.0, fontFamily: "Brand-Bold"),
50                          ),
51                        ),
52                      ],
53                    ),
54                    SizedBox(height: 16.0),
55                    Row(
56                      children: [
57                        Image.asset(
58                          "images/pickicon.png",
59                          height: 16.0,
60                          width: 16.0,
61                        ),
62                        SizedBox(
63                          width: 18.0,
64                        ),
65                        Expanded(
66                          child: Container(
67                            decoration: BoxDecoration(
68                              color: Colors.grey[400],
69                              borderRadius: BorderRadius.circular(5.0),
70                            ),
71                            child: Padding(
```

```
72                                  padding: EdgeInsets.all(3.0),
73                                  child: TextField(
74                                    controller: pickUpTextEditingController,
75                                    decoration: InputDecoration(
76                                      hintText:
77                                          Provider.of<AppData>(context, listen: false)
78                                              .userAddress
79                                              .toString(),
80                                      fillColor: Colors.grey[400],
81                                      filled: true,
82                                      border: InputBorder.none,
83                                      isDense: true,
84                                      contentPadding: EdgeInsets.only(
85                                          left: 11.0, top: 8.0, bottom: 8.0),
86                                    ),
87                                  ),
88                                ),
89                              ),
90                            ),
91                          ],
92                        ),
93                        SizedBox(height: 10.0),
94                        Row(
95                          children: [
96                            Image.asset(
97                              "images/desticon.png",
98                              height: 16.0,
99                              width: 16.0,
100                           ),
101                           SizedBox(
102                             width: 18.0,
103                           ),
104                           Expanded(
105                             child: Container(
106                               decoration: BoxDecoration(
107                                 color: Colors.grey[400],
108                                 borderRadius: BorderRadius.circular(5.0),
109                               ),
110                               child: Padding(
111                                 padding: EdgeInsets.all(3.0),
112                                 child: TextField(
113                                   controller: dropOffEditingController,
114                                   decoration: InputDecoration(
115                                     hintText: "Where to?",
116                                     fillColor: Colors.grey[400],
117                                     filled: true,
118                                     border: InputBorder.none,
119                                     isDense: true,
120                                     contentPadding: EdgeInsets.only(
121                                         left: 11.0, top: 8.0, bottom: 8.0),
122                                   ),
123                                 ),
124                               ),
125                             ),
126                           ),
127                         ],
128                       ),
129                     ],
130                   ),
131                 ),
132               ),
133             ],
134           ),
135         );
136       }
137     }
```

5. The main code

```
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/material.dart';
import 'package:mwslati3/Allscreens/loginScreen.dart';
import 'package:mwslati3/Allscreens/mainscreen.dart';
import 'package:mwslati3/Allscreens/registerationScreen.dart';
import 'package:mwslati3/splashscreen.dart';
import 'package:provider/provider.dart';

import 'DataHandler/appData.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(MyApp());
}

DatabaseReference usersRef =
    FirebaseDatabase.instance.reference().child("users");

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => AppData(),
      child: MaterialApp(
        title: 'Mwslati',
        theme: ThemeData(
          primarySwatch: Colors.blue,
          visualDensity: VisualDensity.adaptivePlatformDensity,
        ),
        initialRoute: SplashScreen.route,
        routes: {
          RegisterationScreen.idScreen: (context) => RegisterationScreen(),
          LoginScreen.idScreen: (context) => LoginScreen(),
          MainScreen.idScreen: (context) => MainScreen(),
          SplashScreen.route: (context) => SplashScreen(),
        },
        debugShowCheckedModeBanner: false,
      ),
    );
  }
}

```