جامعة الأزهر – غـــــزة

Al Azhar University – Gaza

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

FINAL PROJECT REPORT

Prepared by:

Mohammed Jamal Abedellatif 20200565

SUPERVISED BY

Eng. Seraj Sersawi

June.2023

SEMESTER II 2022/2023

Chapter 1 introduction

1.1 Introduction:

In this experiment, we will interface an LCD with a **Raspberry Pi 4**. It is very simple and easy and is commonly used in several electronic products. LCD (Liquid Crystal Display) provides a user-friendly interface and can be very useful for debugging purposes. We will also make a program using **Python** that enable the user to enter a password and check if it is true or not and display the result on the LCD.

1.2 Objectives:

To design a complete circuit that can check if the entered password is correct or not and display the entered password on an LCD.

Chapter 2 Instruments and components

Tools, Instruments and Device:

- Proteus simulation software.
- Raspberry Pi 4.
- SD card, for install operating system.

Components in Proteus simulation:

- Raspberry Pi 4.
- 2*16 LCD.
- 3*4 KEYPAD PHONE.
- BUZZER.
- GREEN LED.
- RED LED.
- NPN TRANSISTOR.
- RELAY CIRCUIT.
- Diode.

Chapter 3 Experiments

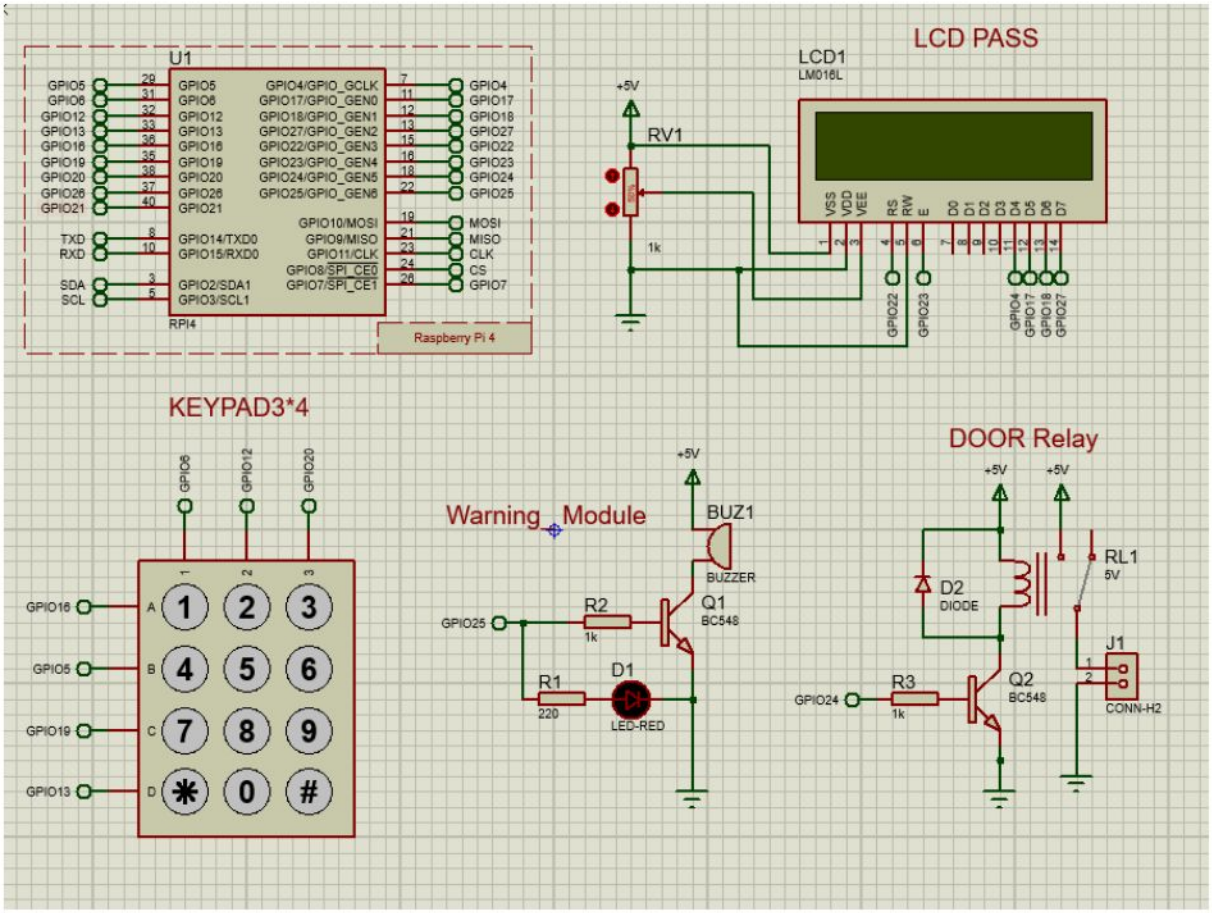
Experimental Procedures:

In this experiment, we are going to make a security system using password. The circuit uses LCD to identify a user who must enter a password to access the system. If the password is correct a Relay Door will be ON which can turn on any applications through relay load contact to show that the password is correct.

Also, if the entry is incorrect three times, a buzzer and red LED will be ON and the system will be break.

Follow up the following steps:

1- Let's first simulate a keypad interfacing circuit using Proteus software. Open Proteus and connect the circuit as shown in Figure below.



1- Using Raspberry Pi 4, first ensure that you have the necessary power supply, SD card, and operating system installed.

2- Connecting the LCD (2*16) display to the Raspberry Pi's GPIO pins as follows:

- LCD_RS to GPIO 22.
- LCD_E to GPIO 23.
- LCD_D4 to GPIO 4.
- LCD_D5 to GPIO 17.
- LCD_D6 to GPIO 18.
- LCD_D7 to GPIO 27.

3- Using potentiometer 10KΩ, for LCD contrast.

4- Connecting the keypad (4*3) to the Raspberry Pi's GPIO pins as follows:

- Connect ROW1 of the keypad to GPIO 16
- Connect ROW2 of the keypad to GPIO 5
- Connect ROW3 of the keypad to GPIO 19
- Connect ROW4 of the keypad to GPIO 13
- Connect COL1 of the keypad to GPIO 6
- Connect COL2 of the keypad to GPIO 12
- Connect COL3 of the keypad to GPIO 20

5- Connecting the Door_Relay output pin to GPIO 24. This pin will control the relay to turn the door ON and OFF.

6- Connecting the Warning_ Module output pin to GPIO 25. This pin will control the warning module to be activate if the password is wrong for 3 time.

7- write the python code in Source Code:

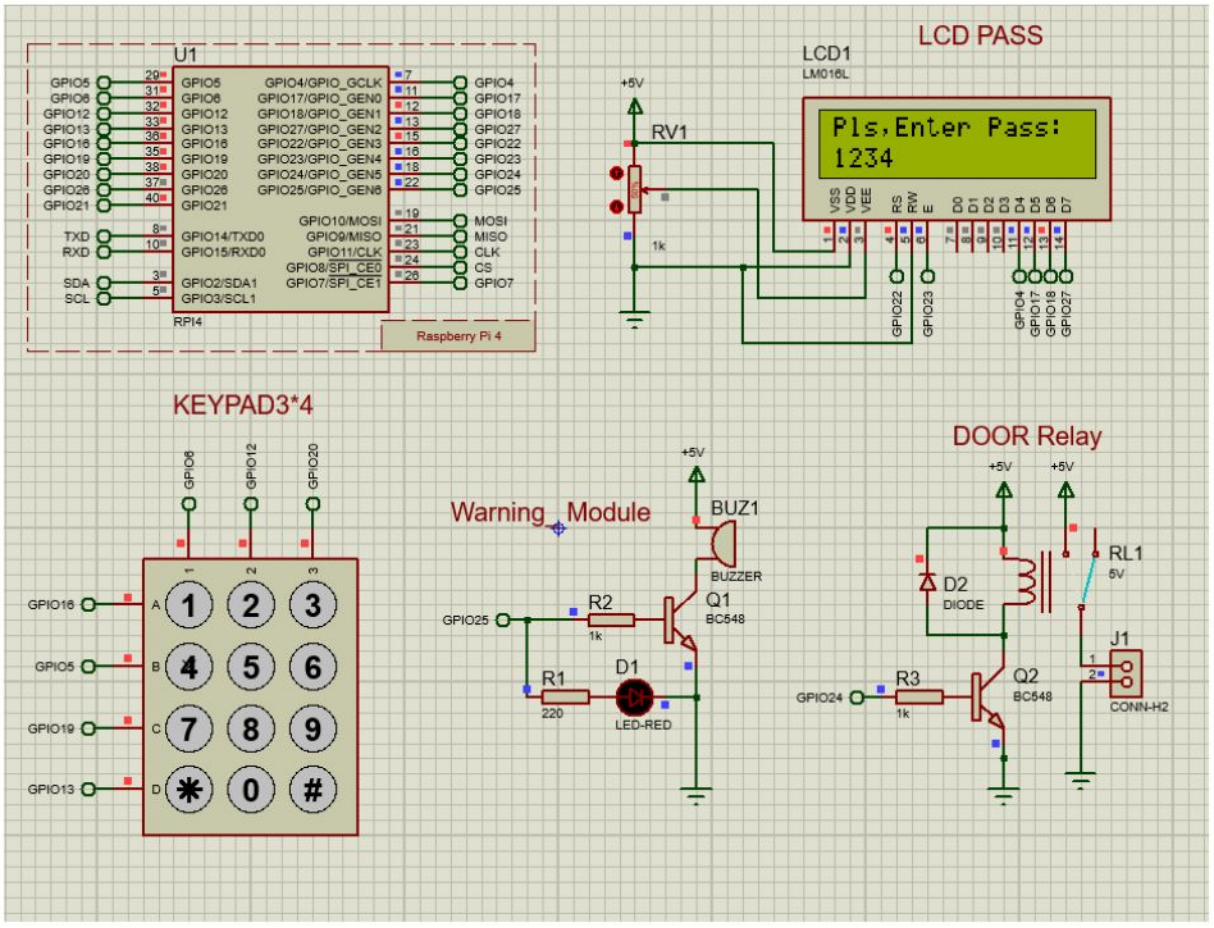
```
import RPi.GPIO as GPIO
from time import sleep

Fan_Relay = 24
Warning_M = 25
Button_Pin = 21
# Define GPIO to LCD mapping
LCD_RS = 22
LCD_E = 23
LCD_D4 = 4
LCD_D5 = 17
LCD_D6 = 18
LCD_D7 = 27
# Define some device constants
LCD_WIDTH = 16  # Maximum characters per line
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0x80  # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0  # LCD RAM address for the 2nd line
# Define keypad matrix
KEYPAD = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
    ["*", 0, "#"]
]

# Define keypad GPIO pins
ROW_PINS = [16, 5, 19, 13]  # Rows 1, 2, 3, 4
COL_PINS = [6, 12, 20]  # Columns 1, 2, 3

# Set GPIO mode and setup keypad pins
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LCD_E, GPIO.OUT)  # E
GPIO.setup(LCD_RS, GPIO.OUT)  # RS
GPIO.setup(LCD_D4, GPIO.OUT)  # DB4
GPIO.setup(LCD_D5, GPIO.OUT)  # DB5
GPIO.setup(LCD_D6, GPIO.OUT)  # DB6
GPIO.setup(LCD_D7, GPIO.OUT)  # DB7
GPIO.setup(Fan_Relay, GPIO.OUT)  # Fan_Relay
GPIO.setup(Warning_M, GPIO.OUT)  # Warning_M]
GPIO.setup(Button_Pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
for pin in ROW_PINS:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.HIGH)
for pin in COL_PINS:
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)


# ===========LCD=============
def lcd_init():
    # Initialise display
    lcd_byte(0x33, LCD_CMD)  # 110011 Initialise
    lcd_byte(0x32, LCD_CMD)  # 110010 Initialise (Set to 4-bit mode)
    lcd_byte(0x06, LCD_CMD)  # 000110 (Cursor move direction)
    lcd_byte(0x0C, LCD_CMD)  # 001100 (Display On,Cursor Off, Blink Off)
    lcd_byte(0x28, LCD_CMD)  # 101000 (Data length, number of lines, font
```

```
size)
    lcd_byte(0x01, LCD_CMD)  # 000001 (Clear display)
    sleep(0.0005)


def lcd_byte(bits, mode):
    GPIO.output(LCD_RS, mode)  # RS
    # High bits
    GPIO.output(LCD_D4, False)
    GPIO.output(LCD_D5, False)
    GPIO.output(LCD_D6, False)
    GPIO.output(LCD_D7, False)
    if bits & 0x10 == 0x10:
        GPIO.output(LCD_D4, True)
    if bits & 0x20 == 0x20:
        GPIO.output(LCD_D5, True)
    if bits & 0x40 == 0x40:
        GPIO.output(LCD_D6, True)
    if bits & 0x80 == 0x80:
        GPIO.output(LCD_D7, True)
    # Toggle 'Enable' pin
    lcd_toggle_enable()
    # Low bits
    GPIO.output(LCD_D4, False)
    GPIO.output(LCD_D5, False)
    GPIO.output(LCD_D6, False)
    GPIO.output(LCD_D7, False)
    if bits & 0x01 == 0x01:
        GPIO.output(LCD_D4, True)
    if bits & 0x02 == 0x02:
        GPIO.output(LCD_D5, True)
    if bits & 0x04 == 0x04:
        GPIO.output(LCD_D6, True)
    if bits & 0x08 == 0x08:
        GPIO.output(LCD_D7, True)
    # Toggle 'Enable' pin
    lcd_toggle_enable()


def lcd_toggle_enable():
    # Toggle enable
    sleep(0.0005)
    GPIO.output(LCD_E, True)
    sleep(0.0005)
    GPIO.output(LCD_E, False)
    sleep(0.0005)


def lcd_string(message, line):
    # Send string to display
    message = message.ljust(LCD_WIDTH, " ")
    lcd_byte(line, LCD_CMD)
    for i in range(LCD_WIDTH):
        lcd_byte(ord(message[i]), LCD_CHR)


# Function to read the keypad
def read_keypad():
    # Scan rows for key press
    for i, row_pin in enumerate(ROW_PINS):
        GPIO.output(row_pin, GPIO.LOW)
```

```
        for j, col_pin in enumerate(COL_PINS):
            if GPIO.input(col_pin) == GPIO.LOW:
                while GPIO.input(col_pin) == GPIO.LOW:
                    pass  # Wait for key release
                GPIO.output(row_pin, GPIO.HIGH)
                return KEYPAD[i][j]
        GPIO.output(row_pin, GPIO.HIGH)

    return None


# Main function
def main():
    lcd_init()
    lcd_string("HI....", LCD_LINE_1)
    sleep(2)
    lcd_string("", LCD_LINE_1)  # Clear line 2 initially
    code = [1, 2, 3, 4]  # Code to match against
    sequence = []  # Entered sequence
    attempts = 0  # Failed attempts counter
    lcd_string("", LCD_LINE_2)  # Clear line 2 initially
    cursor_position = 0  # Initialize cursor position
    try:
        while True:
            lcd_string("Pls,Enter Pass:", LCD_LINE_1)
            key = read_keypad()
            if key is not None:
                sequence.append(key)
                print("Pressed:", key)
                lcd_byte(0xC0 + cursor_position, LCD_CMD)  # Set cursor to
current position
                lcd_byte(ord(str(key)), LCD_CHR)  # Print the digit at the
current position
                cursor_position += 1  # Move cursor to the right
                if cursor_position >= LCD_WIDTH:  # Wrap around to the
beginning if end of line reached
                    cursor_position = 0
                sleep(0.5)
                if len(sequence) == 4:
                    if sequence == code:
                        lcd_string("", LCD_LINE_2)  # Clear line 2
initially
                        print("Welcome")
                        GPIO.output(Fan_Relay, 1)
                        lcd_string("Welcome ^_^", LCD_LINE_1)
                        attempts = 0  # Reset failed attempts counter
                        sleep(4)
                        lcd_string("", LCD_LINE_1)  # Clear line 2
initially
                        lcd_byte(0xC0, LCD_CMD)  # Set cursor to current
position
                        cursor_position = 0
                    else:
                        attempts += 1
                        if attempts == 3:
                            lcd_string("", LCD_LINE_2)  # Clear line 2
initially
                            print("Warning: Maximum attempts reached!")
                            GPIO.output(Warning_M, 1)
                            lcd_string("Warning!!!!!", LCD_LINE_1)
                            break
```

```
                        else:
                            lcd_string("", LCD_LINE_2)  # Clear line 2
initially
                            print("Wrong Pass!!")
                            GPIO.output(Fan_Relay, 0)
                            lcd_string("Wrong Pass!!", LCD_LINE_1)
                            sleep(4)
                            lcd_string("", LCD_LINE_1)  # Clear line 2
initially
                            lcd_byte(0xC0, LCD_CMD)  # Set cursor to
current position
                            cursor_position = 0
                    sequence = []  # Reset sequence
            sleep(0.1)
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()


if __name__ == '__main__':
    main()
```

8- Building project, then Run and test the code as shown below:



2- After simulation making the experiment practical, the same connecting GPIO pins from the Raspberry pi, then open Thonny program in the OP of RPi, to upload the code.