

4.1G

253300L

Danica Fiteni

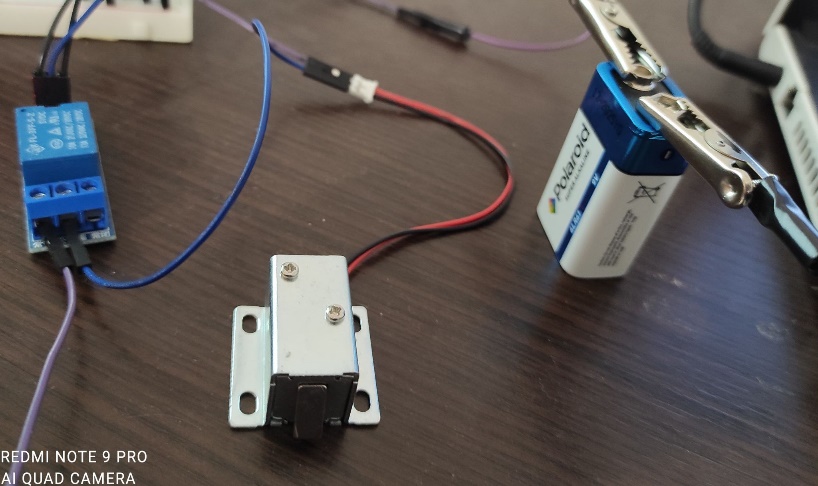
**Embedded System Assignment: Task 4**

**Section 1: : Link to WordPress website showing progress of your project :**

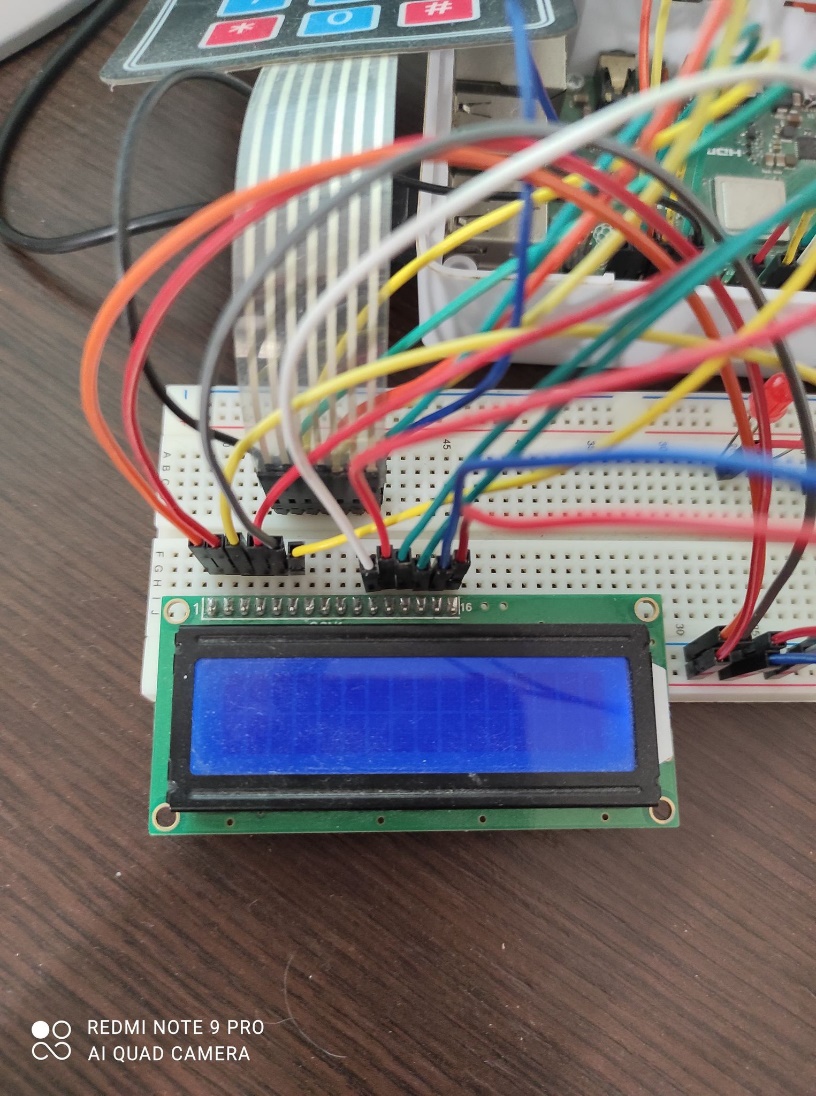
* <https://securebox345538713.wordpress.com/2021/04/12/secure-box/?fbclid=IwAR2x2FnejEYw1yrZqLuJSr0afNYdDtHEPK9PJgG_gcljvfry9ynzcexu5tg>

**Section 2: List and describe the advanced features you used in this project:**

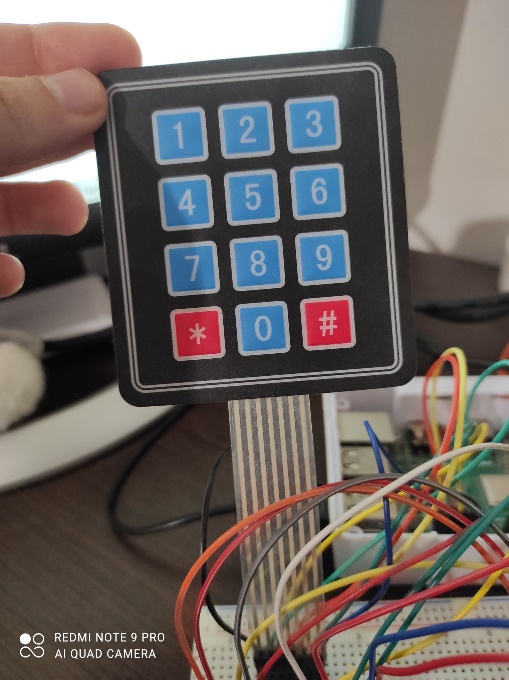
* Solenoid Lock: To made the solenoid work, I connected it with a relay and a 9V battery. When the user enters a correct pin the lock will unlock.



* LCD Screen: the lcd screen is connected with the keypad. If the user enter number 4 on the keypad, it will display it on the lcd screen.

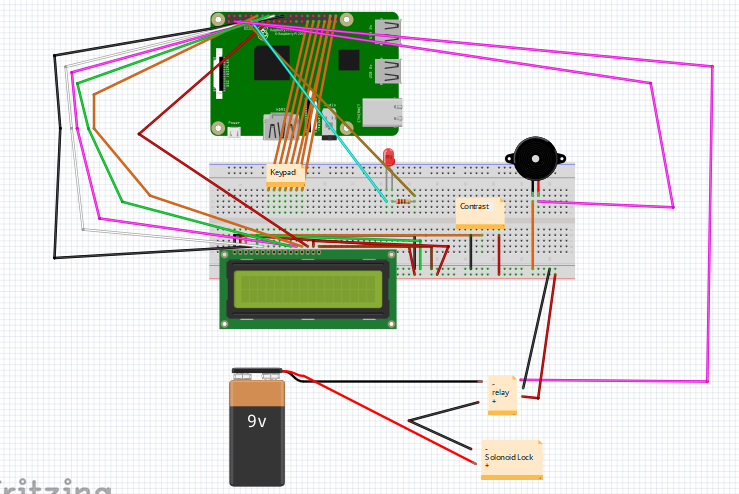


* Keypad: To enter the pin



* Email Alert: If the user enter a correct pin, the secure box will open the lock so it will advise the user that the secure box is open.

**Section 3: Hardware component diagram of your project using Fritzing :**



**Section 4: Mention 5 hardware or software challenges encountered during the duration of the project and mention how these were solved:**

1. Matrix keypad: I was getting a problem when I press number 1 it shows me that I pressed number 4. I was so confused at first but then I realised that I have a 3x4 keypad matrix and I only have 3 columns not 4 columns.
2. Lcd screen: I was very tired of how can I connect the keypad with the lcd screen. I almost gave up but after a lot of research, I have managed to make it work.
3. For the buzzer and led light, I managed to do it easily.
4. To do the email alert it took me about a week to do it but at least I managed
5. For the relay and solenoid, I found it very difficult. My problem was that when I connected it with pin 3 it did not worked. After a while I connected it with pin 16 and it worked.

**Section 5: A copy of your entire project Python code :**

import RPi.GPIO as GPIO

import time

from pad4pi import rpi\_gpio

from gpiozero import LED

from time import sleep

import smtplib

import RPi.GPIO as GPIO

GPIO.setwarnings(False)

led = LED(4)

lock\_pin = 16

GPIO.setmode(GPIO.BCM)

buzzer = 3

GPIO.setup(buzzer, GPIO.OUT)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

KEYPAD = [

["1", "2", "3"],

["4", "5", "6"],

["7", "8", "9"],

["\*", "0", "#"]

]

COL\_PINS = [26,20,21] # BCM numbering

ROW\_PINS = [5, 6,13,19] # BCM numbering

factory = rpi\_gpio.KeypadFactory()

keypad = factory.create\_keypad(keypad=KEYPAD, row\_pins=ROW\_PINS, col\_pins=COL\_PINS)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

def printKey(key):

lcd\_byte(ord(key),LCD\_CHR)

if key == "#":

lcd\_string(" Correct Pin :) ",LCD\_LINE\_1)

time.sleep(1)

lcd\_string(" Unlocked", LCD\_LINE\_2)

lcd\_byte(0xC0, LCD\_CMD)

led.on()

#email alert

gmail\_user = "danfit784@gmail.com"

gmail\_password = "tssphpbqjnfschti"

sent\_from = gmail\_user

to = ["danicafiteni@gmail.com"]

subject = 'OMG Super Important Message'

body = 'Hey, your secure box is unlocked :). Well Done.'

email\_text = """\

From: %s

To: %s

Subject: %s

%s

""" % (sent\_from, ", ".join(to), subject, body)

try:

server = smtplib.SMTP\_SSL('smtp.gmail.com', 465)

server.ehlo()

server.login(gmail\_user, gmail\_password)

server.sendmail(sent\_from, to, email\_text)

server.close()

print( 'Email sent!')

except:

print('Something went wrong...Email not sent')

time.sleep(10)

led.off()

lcd\_string(" ",LCD\_LINE\_1)

lcd\_string(" ",LCD\_LINE\_2)

print("Unlocking")

GPIO.setup(lock\_pin, GPIO.OUT)

GPIO.output(lock\_pin, GPIO.HIGH)

else:

lcd\_string("Incorrect Pin :) ",LCD\_LINE\_1)

lcd\_string(" Try again ", LCD\_LINE\_2)

led.off()

GPIO.output(buzzer, GPIO.HIGH)

sleep(0.5)

GPIO.output(buzzer, GPIO.LOW)

sleep(0.5)

lcd\_string("Enter your pin :", LCD\_LINE\_1)

lcd\_string(" ", LCD\_LINE\_2

)

lcd\_byte(0xC0, LCD\_CMD)

GPIO.setup(lock\_pin, GPIO.OUT)

GPIO.output(lock\_pin, GPIO.LOW)

GPIO.cleanup(lock\_pin)

print("Lock stay locked")

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

# printKey will be called each time a keypad button is pressed

keypad.registerKeyPressHandler(printKey)

# Define GPIO to LCD mapping

LCD\_RS = 25

LCD\_E = 24

LCD\_D4 = 23

LCD\_D5 = 17

LCD\_D6 = 18

LCD\_D7 = 22

# Define LCD parameters

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

# Timing constants

E\_PULSE = 0.0005

E\_DELAY = 0.0005

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

def main():

# Main program block

global pm

global system\_sts

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM) # Use BCM GPIO numbers

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

# Initialise display

lcd\_init()

lcd\_byte(0x01, LCD\_CMD)

lcd\_string("Enter your pin : ",LCD\_LINE\_1)

lcd\_byte(0xC0, LCD\_CMD)

while True:

time.sleep(1)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

def lcd\_init():

# Initialise display

lcd\_byte(0x33,LCD\_CMD) # 110011 Initialise

lcd\_byte(0x32,LCD\_CMD) # 110010 Initialise

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

time.sleep(E\_DELAY)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

def lcd\_byte(bits, mode):

# Send byte to data pins

# bits = data

# mode = True for character

# False for command

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

time.sleep(E\_DELAY)

GPIO.output(LCD\_E, True)

time.sleep(E\_PULSE)

GPIO.output(LCD\_E, False)

time.sleep(E\_DELAY)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

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)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

if \_\_name\_\_ == '\_\_main\_\_':

try:

main()

except KeyboardInterrupt:

pass

finally:

lcd\_byte(0x01, LCD\_CMD)

lcd\_string("Goodbye!",LCD\_LINE\_1) #CTRL + C

GPIO.cleanup()