www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 7 (July 2013), Page No. 2235-2257

DESIGN AND CONSTRUCTION OF DOOR LOCKING SECURITY SYSTEM USING GSM

Ushie James Ogri, Donatus Enang Bassey Okwong, Akaiso Etim

Department of Physics, University of Calabar,
ushjames@yahoo.com

ABSTRACT

This project presents a prototype security door that can be remotely controlled by a GSM phone set acting as the transmitter and another GSM phone set with a dual tone multi-frequency (DTMF) connected to the door motor through a DTMF decoder interfaced with microcontroller unit and a stepper motor .The design is composed of four main functional modules, namely; the GSM module, the decoding module, controlling module and the switching module. The GSM module act as both transmitting and receiving unit employs the use of a mobile phone set serving as the communication device between the user at one end and the object of access (i.e. the door) at the other receiving end. The decoding module and the controlling module are made possible using modern integrated circuit chips ensuring proper conversion of signal to binary codes, enabling the microcontroller to communicate properly with the switching device responsible for opening and closing the door. The codes for this project was written in assembly language with Visual basic software and compiled with M-IDE studio for MC-51compiler which work perfectly with Window XP environment, the program run without error before it was burn onto the microcontroller using a device called the programmer by placing the microcontroller on it socket equal to the pin number.

Keywords: Door Locking, Security, GSM, Microcontroller and Stepper Motor

INTRODUCTION: Security describes protection of life and property. There are doors to keep people out, Key locks and chains reinforce the mode of security. Doors are being made of metals not just wood anymore. Influential persons in our society have bullet proof doors to ensure a good measure of security of self and family. The security sector is experiencing diversification as it has never seen before. This has brought about the need to review the reliability of already existing systems and look into the possibility of creating better systems that are smarter and more secure.

The micro controller based digital lock presented here is an access control system that allows only authorized persons to access a restricted area, this system is best suitable for corporate offices, automated machine (ATMs) and home security. It comprises of a small electronic unit which is in fixed at the entry door to control a solenoid-operated lock with the help of a stepper motor, when an authorized person enters predetermined user password via the global system for mobile communication (GSM) keypad, the stepper motor is operated for a limited time to unlatch the solenoid-operated lock so the door can be open. At the end of preset delay time, the stepper motor is operated in reverse direction and the door gets locked again.

When the code has been incorrectly entered three times in a row, the code lock will switch to block mode, this function thwarts any attempt by 'hackers' to quickly try a large number of codes in a sequence. If the user forgets his password, the code lock can be accessed by a unique 8 digit administrator password and the secret code can be changed any time after entering the current code (Master code).

The project intends to interface the microcontroller with the GSM modem and start/stop the engine by sending the predefined messages from the mobile phone to the controlling unit, The software application and the hardware implementation help the microcontroller read the messages sent by the user from a mobile phone or send messages to the mobile phone through the modem and accordingly change the status of the engine motor required. The measure of efficiency is based on how fast the microcontroller can detect the incoming message and act accordingly.

The system is totally designed using GSM and embedded systems technology. The Controlling unit has an application program to allow the microcontroller read the incoming data through the modem and control the engine motor as per the requirement. The performance of the design is maintained by the controlling unit.

This project uses 8051 microcontroller as the central processing unit. Specifically the proto-type make used of AT89s52 microcontroller with Programs written in assembly language burnt inside the microcontroller to perform the following capabilities;

Assembly language is used to write the interfacing program and compiled with M-IDE studio for MC-51compiler which work perfectly with Window XP environment and may have compatibility problems with higher versions of the Window operating system

In residential applications: solid wood door, panel doors, metal skinned wood-edged doors and metal edge-wrapped doors (www.wikipedia.org, 2008). In addition to doors are; deadbolts, frame reinforcements, door chains and hinge screws – long 3" screws (www.statefarm.com, 2012) but despite these reinforcements door, security by itself is very porous. An electronics or electric lock is a locking device which operates by means of electric current (Gibson Stan, 2001). One of such locks is magnetic locked (mag locked).

A large electro-magnet is mounted on the door frame and a corresponding armature is held fast to the magnet (Mckenice, 1995). mag locks by design fail unlocked, that is if power is removed they unlock.

SYSTEM DESIGN: The design of a door locking security system using GSM is a complex design which comprises of so many modules (parts) brought together to form the overall design. Each of these modules is made up of discrete components that are joined together to achieve a particular purpose. These separate modules are: The Power Supply Unit, The Buzzer Unit, The micro controller Unit, Telephone unit and Switching.

These different units cannot function alone, they all need to function together to achieve the desired result. The GSM modem received tone from the GSM network as shown by the direction of the arrow in the diagram below and transmit same to the DTMF decoder but the current value was very small (i.e. about 0.1mA) it was step-up by the tone transformer so that it could be decode by the DTMF decoder which then send the decoded codes to the microcontroller for processing and outputting to relevant component to act accordingly.

The block diagram of the design showing all the units combined together are shown in the figure below.

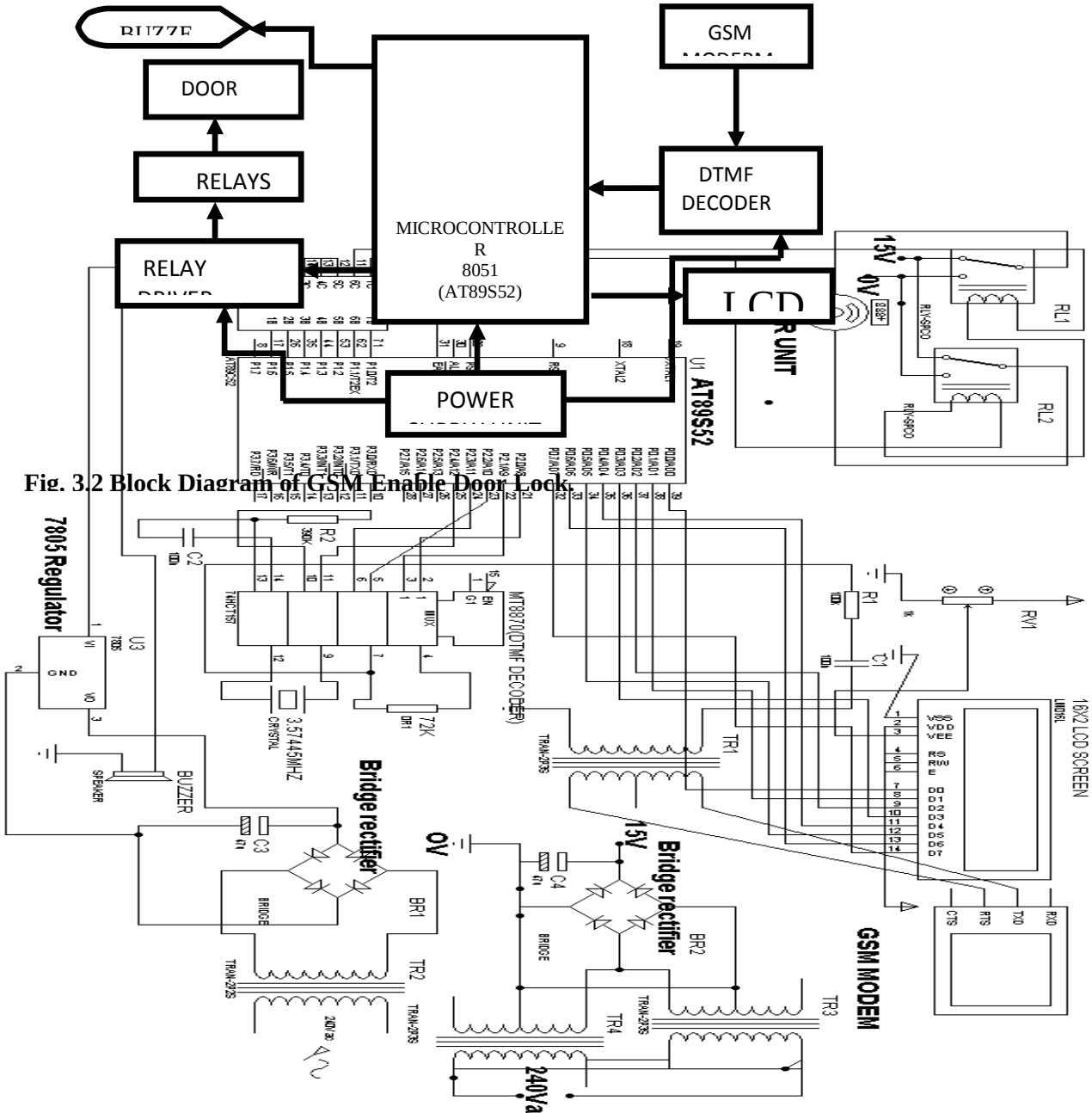


Fig: 3

Fig. 3.2 Block Diagram of GSM Enable Door Lock.

SOFTWARE PROGRAMS FOR THE MICROCONTROLLER: Microcontroller is a programmable device (Mazidi, 1997). It is an intelligent core for a specialised dedicated system (Sanchez & Canton, 2007). The firmware part deals with programming the microcontroller so that it can control the operation of the IC's used in the hardware implementation. In the research, M-IDE studio for MC-51 software development tool is used to compile the source code, which was written in assembly language. The Universal programmer was used to burn the compile source code onto the microcontroller.

Software development involves a series of steps which are necessary for the development of reliable and maintainable software.

SYSTEM FLOW CHART: A flow chart showing in detail the working of thee device is shown below. From this flow chart, we can see how the different unit come together to achieve the desired purpose.

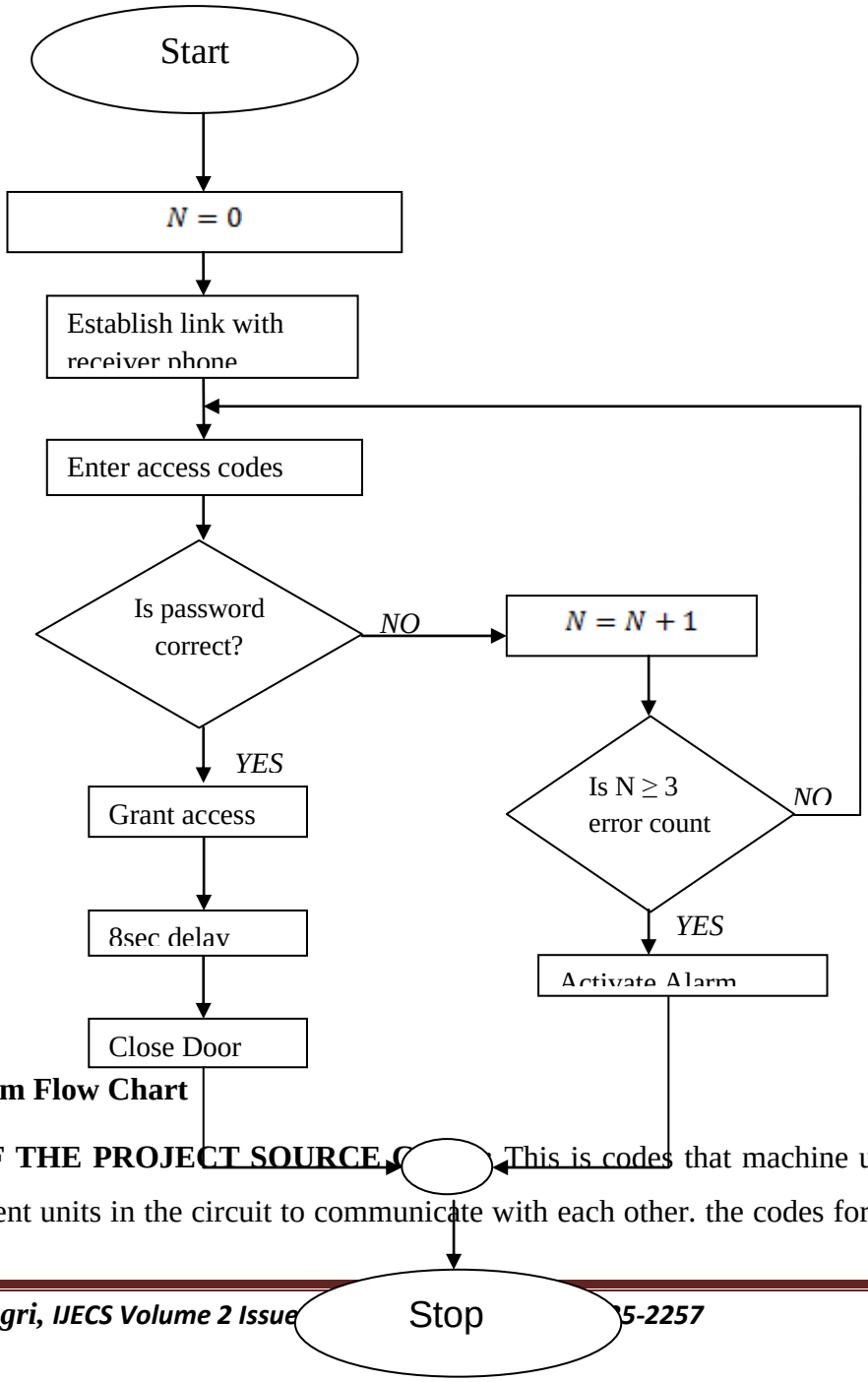


Fig 3.10: System Flow Chart

WRITING OF THE PROJECT SOURCE C This is codes that machine understand which enable all the component units in the circuit to communicate with each other. the codes for this project was written

in assembly language with Visual basic software and compiled with M-IDE studio for MC-51compiler which work perfectly with Window XP environment, the program run without error before it was burn onto the microcontroller using a device called the programmer by placing the microcontroller on it socket equal to the pin number of the microcontroller. The source code is at appendix.

RESULTS AND DISCUSSION: The prototype door security system developed in this project did well in achieving its original goals. In the beginning the system will boot up with display on the LCD screen prompting the user to enter pin code.

The password door lock system has a default password of "198526", 196310 and the user is given only 3 attempts to enter the correct password. If not, the keypad will switch to block mode requesting for PUK number which is "38893982" eight numbers. At the same time an alarm will sound until the PUK number is imputed with correct PIN. The development of this technology for the field of security system is not only possible, but it could even prove to be very useful.

SUMMARY AND WORKING PROCEDURE OF THE PROJECT: The operation of this project is summarized as follow;

i. A call is placed to the phone that is connected to the system, this call is like any normal call to a friend, colleague etc. the call made is set to be automatically answered at the other (i.e. door) end, the caller immediately presses six digits numbers (password).

ii. The signal qualities of the tones are first increased by passing it into a step up transformer, the output of which goes to the DTMF decoder.

iii. In the DTMF decoder the tones are received and decoded into a binary code equivalent, the output of the decoder is sent to the microcontroller.

iv. The microcontroller's internal programming processes the output from the DTMF decoder. Here, these decoded signals are identified as the keys pressed on the phone keypad. the microcontroller output these information into three unit;

- ✓ Liquid crystal display unit, to show the user the digit pressed.
- ✓ The ULN2003 driver. this converts the logic level from the microcontroller's TTL to the signal that control the switching sequence of the relay
- ✓ The Buzzer alarm. This sound to alert the user when a digit is pressed and also sound continuously when wrong numbers are entered by intruder.
- ✓ On entry of the six digit code the "#" button of the keypad is pressed as confirmation of the code. If the code entered is correct, (if the user mistakenly typed wrong digit, this can be delete by pressing "0" key to backspace) data is sent to the microcontroller to activate door opening sequence; this sequence includes the display of an "Access Granted" text on the LCD screen and

the output of a signal to the transistor driving the relay. This signal causes the relay contacts to switch and completes the motor circuit thereby causing the door to open.

- ✓ The door closes automatically after precisely 8 seconds, but user can close the door by pressing the "#" key on the keypad. The microcontroller is programmed to recognized this character and bring about the switching action of another relay which closes the door.

Table 4.1 Component Description and Prices

COMPONENT DESCRIPTION	UNIT PRICE (N)	QUANTITY	TOTAL PRICE (N)
MICRO-CONTROLLER(AT89S52)	1200	1	1200
TONE TRANSFORMER 240/12v	500	1	500
16X2 LCD SCREEN	2000	1	2000
DTMF DECODER	3000	1	3000
3.75445MHZ CRYSTAL OSCILLATOR	100	2	200
30pF CAPACITOR	50	4	200
10µF,16v CAPACITOR	100	1	100
12v/500mA TRANSFORMER	500	1	500
BRIDGE RECTIFY	300	2	600
1000µf,25v CAPACITOR	200	2	200
LM7805 REGULATOR	150	1	150
10K POTENTIOMETER	100	1	100
RESISTOR	20	5	100
12V/500mA TRANSFORMER	500	1	500
VERIO BAORD	200	1	200
SOLDERING IRON	300	1	300
SOLDERING LEAD	500	1	500
150pF CAPACITOR	100	1	100

I.C SOCKET	50	3	150
GSM MODEM	7000	1	7000
12V/10A RELAY	200	2	400
ULN2003 RELAY DRIVER	300	1	300
DOOR FABRICATION AND SERVO MOTOR	7000	1	7000
PROGRAMMING LOGISTICS	15000	-	15000
TRANSPORTATION AND MISCELENOUS	10000	-	10000
15v/2000mA TRANSFORMER	500	2	1000
EAR PIECE,13A PLUG & CONNECTORS	1400	1	1400
GRAND TOTAL	**-**	**-**	**52,000**

CONCLUSION

The work was successful. It is evidence that the use of mobile phones with the right circuitry can be used to operate a security system, since the mobile phone in today's world; it is an access device a lot easier and affordable to obtain as opposed to specially fabricated keys and smart-cards. The ability of the system to accesses a secure place (Home, office, ATM etc.) remotely almost anywhere in the world is a plus since technology has made the world a global village.

REFRENCES

AT89s52 datasheet, www.atmel.com, (13 December, 2012).

Bill Bowden, "controlling Relay with logic signals" http//ourworld.compuserve.com, (10 September, 2008).

Crystal Oscillator, http://en.wikipedia.org (10th September 2011).

Door Hinges and security, http://www.statefarm.com/learningbsafeathmburghing.asp (20th February, 2008).

Electric locks, http://en.wikipedia.org/wiki/door security, (10th November, 2012).

Gibson Stan (2001) "the illustrated Dictionary of electronics", McGraw-Hill Publication, USA.

Horowitz, Paul; Winfield, H. (1995) "The Art of Electronics", Cambridge University Press, London.

Mckenice Smith Ian, Hughes (1995) "Electrical technology" 7th edition, Longman group ltd New York.

MT8070D1 datasheet, www.mitel.com (11th November, 2011).

Muhammad Ali Mazidi, Janice Gillispie, Mazidi Rolin and D. McKinlayn (1997) "The 8051 Microcontroller and Embedded Systems Using Assembly and C" 2nd Edition, Dept. of Computer Science and Information Engineering, National Cheng Kung University, TAIWAN.

Sanchez Julio and Canton Maria (2007), "Microcontroller programming- the micro chip PIC", CRC Press USA.

APPENDIX A:
PROJECT SOURCE CODE

```
org 00h ; reset vector address
Data_Ram_0 data 30
Data_Ram_1 data  Data_Ram_0 + 1
Data_Ram_2 data  Data_Ram_1 + 1
Data_Ram_3 data  Data_Ram_2 + 1
Data_Ram_4 data  Data_Ram_3 + 1
Data_Ram_5 data  Data_Ram_4 + 1
Data_Ram_6 data  Data_Ram_5 + 1
Data_Ram_7 data  Data_Ram_6 + 1
Data_Ram_8 data  Data_Ram_7 + 1
Data_Ram_9 data  Data_Ram_8 + 1
      receive_bit equ P1.0
      DTMF_receive_bitQA equ P1.4
      DTMF_receive_bitQB equ P1.3
      DTMF_receive_bitQC equ P1.2
      DTMF_receive_bitQD equ P1.1
data_bank data 20
rs bit p2.7
rw bit p2.6
en bit p2.5
sdata data p3
 ADC_Data data  p1
 ADC_clock bit p0.4
 relaya bit p2.0
  relayb bit p2.1
 buzzer bit p0.2
 bank data 49
clr relaya
```

```
clr relayb
clr buzzer
mov r0 , #Data_Ram_9
    hat1:     mov   @r0  , #' '
    dec r0
        cjne r0 , #Data_Ram_0 -1 , hat1
clr relaya
clr relayb
clr buzzer
mov r7 , #0
setb rw
                clr en
                setb en
                lcall clear_lcd
                lcall init_lcd
                lcall clear_lcd
                clr rs
                mov sdata,#80h+00h
                setb en
                clr en
                lcall wait_lcd
                                mov Dptr,#message1
Repeat_Data_processingxx : call wait
                                loop212: clr a
                                movc a , @a+Dptr
                                inc Dptr
                                  cjne a,#'@' , jaj212
                                  clr rs
                MOV SDATA,#80H+40H
                SETB EN
                CLR EN
                LCALL WAIT_LCD
                jmp     Repeat_Data_processingxx
                jaj213w: call write_text
 jmp    Repeat_Data_processingxx
        jaj212 :cjne a,#'#' , jaj213w
        call wait
        call wait
        call wait
        call wait
        setb rw
                clr en
                setb en
                lcall clear_lcd
                lcall init_lcd
                lcall clear_lcd
                clr rs
                mov sdata,#80h+00h
                setb en
                clr en
                lcall wait_lcd
```

```
                                        mov Dptr,#message2
Repeat_Data_processingxx11 : call wait

                                loop2121: clr a
                                movc a , @a+Dptr
                                inc Dptr
                 jaj213:                cjne a,#'@' , jaj21211n
                                  clr rs
                        MOV SDATA,#80H+40H
                        SETB EN
                        CLR EN
                        LCALL WAIT_LCD
                        jmp Repeat_Data_processingxx11
                                jaj21211n:cjne a,#'#' , jaj213z
                call wait
                call wait
                call wait
                call wait
                setb rw
                        clr en
                        setb en
                        lcall clear_lcd
                        lcall init_lcd
                        lcall clear_lcd
                        clr rs
                        mov sdata,#80h+00h
                        setb en
                        clr en
                        lcall wait_lcd
                                                mov Dptr,#message3
                        jmp james
 jaj213z : call write_text
  jmp    Repeat_Data_processingxx11
           james :     call wait
                                loop2121c: clr a
                                movc a , @a+Dptr
                                inc Dptr
                                  cjne a,#'#' , jaj21211nc
                                          CALL prompting
                                 sjmp start_validation
           jaj21211nc:  call write_text
           sjmp james
          start_validation :
          jnb        receive_bit , $
          setb buzzer
call       DTMF_DECODER_READER2
call wait
clr buzzer
           jb        receive_bit , $
           jmp     start_validation
 wait_lcd:
```

```
                clr en ; rt lcd command
                clr rs ;it's a command
                setb rw ;it's a read command
                mov sdata,#0ffh ;set all pins to ff initially
                setb en ;clock out command to lcd
                mov a,sdata ;read the return value
                jb acc.7,wait_lcd ;if bit 7 high, lcd still busy
                clr en ;finish the command
                clr rw ;turn off rw for future commands
                ret
init_lcd:
                clr rs
                mov sdata,#38h
                setb en
                clr en
                lcall wait_lcd
                clr rs
                mov sdata,#0eh
                setb en
                clr en
                lcall wait_lcd
                clr rs
                mov sdata,#06h
                setb en
                clr en
                lcall wait_lcd
                ret
clear_lcd:
                clr rs
                mov sdata,#01h
                setb en
                clr en
                lcall wait_lcd
                ret
write_text:

                setb rs
                mov sdata,a
                setb en
                clr en
                lcall wait_lcd
                ret
waitx:

TT0c:  MOV   R3,#8
    MOV   R2,#8
    MOV   R1,#236
TT1c:  DJNZ  R1,TT1c
    DJNZ  R2,TT1c
    DJNZ  R3,TT1c
    RET
```

```
ret
DTMF_DECODER_READER2:
	;scanning for button one 1==0001
	jnb DTMF_receive_bitQA ,ExitSubB0
	jb DTMF_receive_bitQB ,ExitSubB0
	jb DTMF_receive_bitQC ,ExitSubB0
	jb DTMF_receive_bitQD ,ExitSubB0
mov  a , #'*'
	call write_text
mov data_bank, #'1'
CALL SHIFT_DATA
	ret
	ExitSubB0:;scanning for button two 2==0010
	jb DTMF_receive_bitQA ,ExitSubBB
	jnb DTMF_receive_bitQB ,ExitSubBB
	jb DTMF_receive_bitQC ,ExitSubBB
	jb DTMF_receive_bitQD ,ExitSubBB
	mov  a , #'*'
	call write_text
mov data_bank, #'2'
CALL SHIFT_DATA
	;call play2
	ret
	;;;;;;;;;;;;;;;;;;
	ExitSubBB:;scanning for button THREE 3==0011
	jNb DTMF_receive_bitQA ,ExitSubBC
	jNb DTMF_receive_bitQB ,ExitSubBC
	jb DTMF_receive_bitQC ,ExitSubBC
	jb DTMF_receive_bitQD ,ExitSubBC
	mov  a , #'*'
	call write_text
mov data_bank, #'3'
CALL SHIFT_DATA
	ret
	ExitSubBC:
	;;;;;;;;;;;;;;;;
	;scanning for button four 4==0100
	jb DTMF_receive_bitQA ,ExitSu
	jb DTMF_receive_bitQB ,ExitSu
	jnb DTMF_receive_bitQC ,ExitSu
	jb DTMF_receive_bitQD ,ExitSu
	mov  a , #'*'
	call write_text
mov data_bank , #'4'
CALL SHIFT_DATA
	ret
	;;;
	ExitSu:;scanning for button five 5==0101
	jNb DTMF_receive_bitQA ,ExitSu1
	jb DTMF_receive_bitQB ,ExitSu1
	jnb DTMF_receive_bitQC ,ExitSu1
```

```
        jb DTMF_receive_bitQD ,ExitSu1
        mov  a , #'*'
        call write_text
mov data_bank, #'5'
CALL SHIFT_DATA
        ret
        ;;;;%%%%%%%%%%%%%%%%%%%%%%
        ExitSu1:;scanning for button six 6==0110
        jb DTMF_receive_bitQA ,Exit
        jNb DTMF_receive_bitQB ,Exit
        jnb DTMF_receive_bitQC ,Exit
        jb DTMF_receive_bitQD ,Exit
        mov  a , #'*'
        call write_text
mov data_bank, #'6'
CALL SHIFT_DATA
        ret
        Exit:;scanning for button7==0111
        jNb DTMF_receive_bitQA ,Exit1
        jNb DTMF_receive_bitQB ,Exit1
        jNb DTMF_receive_bitQC ,Exit1
        jb DTMF_receive_bitQD ,Exit1
        mov  a , #'*'
        call write_text
mov data_bank, #'7'
CALL SHIFT_DATA
        ret
        Exit1:;scanning for button8==1000
        jb DTMF_receive_bitQA ,ExitX
        jb DTMF_receive_bitQB ,ExitX
        jb DTMF_receive_bitQC ,ExitX
        jNb DTMF_receive_bitQD ,ExitX
        mov  a , #'*'
        call write_text
mov data_bank, #'8'
CALL SHIFT_DATA
        ret
        ExitX:;scanning for button9==1001
        jNb DTMF_receive_bitQA ,ExitA1
        jb DTMF_receive_bitQB ,ExitA1
        jb DTMF_receive_bitQC ,ExitA1
        jNb DTMF_receive_bitQD ,ExitA1
        mov  a , #'*'
        call write_text
mov data_bank, #'9'
CALL SHIFT_DATA
        ret
        ExitA1:;scanning for button*==1011
        jNb DTMF_receive_bitQA ,ExitXX1
        jNb DTMF_receive_bitQB ,ExitXX1
        jb DTMF_receive_bitQC ,ExitXX1
```

```
       jNb DTMF_receive_bitQD ,ExitXX1
        call delete_data_process
              ret
       ;;;&&&&&&&&&&&&&&&&&&&&
       ExitXX1:;scanning for button0==1010
       jb DTMF_receive_bitQA ,ExitXXX1
       jNb DTMF_receive_bitQB ,ExitXXX1
       jb DTMF_receive_bitQC ,ExitXXX1
       jNb DTMF_receive_bitQD ,ExitXXX1
       call delete_data_process
       ret
       ExitXXX1:;scanning for button#==1100
       jb DTMF_receive_bitQA ,ExitXXXX1_error
       jb DTMF_receive_bitQB ,ExitXXXX1_error
       jNb DTMF_receive_bitQC ,ExitXXXX1_error
       jNb DTMF_receive_bitQD ,ExitXXXX1_error
          call verify
       ret
ExitXXXX1_error :       ret
     SHIFT_DATA:
          mov Data_Ram_9 ,Data_Ram_8
                mov Data_Ram_8 ,Data_Ram_7
                      mov Data_Ram_7 ,Data_Ram_6
                            mov Data_Ram_6 ,Data_Ram_5
                                  mov Data_Ram_5 ,Data_Ram_4
                                  mov Data_Ram_4 ,Data_Ram_3
mov Data_Ram_3 ,Data_Ram_2
  mov Data_Ram_2,Data_Ram_1
mov Data_Ram_1 ,Data_Ram_0
 mov Data_Ram_0 ,data_bank
     ret
     verify:
     mov r0 ,#Data_Ram_9
    ;password_1 : db '198526#'
;password_2 : db '196310#'
cjne  @r0 ,#' ' , next
      mov  a , @r0

     dec r0

cjne  @r0 ,#' ' , next
      mov  a , @r0

     dec r0
cjne  @r0 ,#' ' , next
      mov  a , @r0
     dec r0
cjne  @r0 ,#' ' , next
      mov  a , @r0
     dec r0
cjne  @r0 ,#'2' , next
```

```
      mov  a , @r0
     dec r0
      cjne  @r0 ,#'9' , next
      mov  a , @r0
     dec r0
       cjne  @r0 ,#'6' , next
       mov  a , @r0
     dec r0
cjne  @r0 ,#'3' , next
mov  a , @r0
      call write_text
     dec r0
            cjne  @r0 ,#'2' , next
            mov  a , @r0
      call write_text
     dec r0
cjne  @r0 ,#'6' , next
mov  a , @r0
      call write_text
     dec r0
        call open
        ret
        ;password_1 : db '198526#'
next:         mov r0 ,#Data_Ram_9
cjne  @r0 ,#' ' , next1
      mov  a , @r0
      call write_text
     dec r0
cjne  @r0 ,#' ' , next1
      mov  a , @r0
      call write_text
     dec r0
cjne  @r0 ,#' ' , next1
      mov  a , @r0
      call write_text
     dec r0
cjne  @r0 ,#' ' , next1
      mov  a , @r0
      call write_text
     dec r0
 cjne  @r0 ,#'2' , next1
     dec r0
      cjne  @r0 ,#'9' , next1
     dec r0
       cjne  @r0 ,#'8' , next1
     dec r0
cjne  @r0 ,#'5' , next1
     dec r0
            cjne  @r0 ,#'2' , next1
            dec r0
cjne  @r0 ,#'6' , next1
```

```
      dec r0
         call open

         ret
next1:     inc  r7
 cjne r7 , #3 , MAM1
 jmp sat
 mam1: jmp mam
sat:
 CALL PUK
 mov r7 , #0
    mov  DPTR , #unlock
    call ogba
    Repeat_Data_processingxx41 :
                                    loop2124: clr a
                                    movc a , @a+Dptr
                                    inc Dptr
                                      cjne a,#'@' , jaj2124
                                      clr rs
                    MOV SDATA,#80H+40H
                    SETB EN
                    CLR EN
                    LCALL WAIT_LCD
                    jmp      Repeat_Data_processingxx41
                    jaj213w4: call write_text
 jmp     Repeat_Data_processingxx41
          jaj2124 :cjne a,#'#' , jaj213w4
  mov r0 ,#Data_Ram_0
   nextg :   mov  @r0  , #' '
         inc r0
       cjne  r0 ,#Data_Ram_9 + 1   , nextg
       gagg:
 jnb   receive_bit , $
        ExitXXX1a:;scanning for button#==1100
       jb DTMF_receive_bitQA ,ExitXXXX1_errora
       jb DTMF_receive_bitQB ,ExitXXXX1_errora
       jNb DTMF_receive_bitQC ,ExitXXXX1_errora
       jNb DTMF_receive_bitQD ,ExitXXXX1_errora
       ; 34493941
      mov r0 ,#Data_Ram_9
      cjne  @r0 ,#' ' , nextl2
      dec r0
      cjne  @r0 ,#' ' , nextl2
      dec r0
       cjne  @r0 ,#'3' , nextl2
      dec r0
        cjne  @r0 ,#'8' , nextl2
       dec r0
             cjne  @r0 ,#'8' , nextl2
      dec r0
cjne  @r0 ,#'9' , nextl2
```

```
     dec r0
          cjne  @r0 ,#'3' , nextl2
     dec r0
      cjne  @r0 ,#'9' , nextl2
     dec r0
       cjne  @r0 ,#'8' , nextl2
       dec r0
       cjne  @r0 ,#'2' , nextl2
     dec r0
       clr buzzer
       mov r0 ,#Data_Ram_0
  nextgz:   mov  @r0  , #' '
     inc r0
      cjne  r0 ,#Data_Ram_9 + 1   , nextgz
call prompting
       ret
        nextl2:mov r0 ,#Data_Ram_9
  nextgd :   mov  @r0  , #' '
        dec r0
      cjne  r0 ,#Data_Ram_0 - 1   , nextgd
       ExitXXXX1_errora :
  call DTMF_DECODER_READER2
   jb    receive_bit  , $
   jmp gagg
MAM:  call errorr
mov r0 , #Data_Ram_9
 nextgdc:   mov  @r0  , #' '
        dec r0
      cjne  r0 ,#Data_Ram_0 - 1   , nextgdc
call wait
call wait
call wait
RET
OPEN:
setb rw
                    clr en
                    setb en
                    lcall clear_lcd
                    lcall init_lcd
                    lcall clear_lcd
                    clr rs
                    mov sdata,#80h+00h
                    setb en
                    clr en
                    lcall wait_lcd
                    mov r0 , #Data_Ram_9
 nextgdca:   mov  @r0  , #' '
        dec r0
      cjne  r0 ,#Data_Ram_0 - 1   , nextgdca
clr buzzer
MOV DPTR , #access
```

```
AGAIN:
Repeat_Data_processingxxn:
                                loop212J: clr a

                                movc a , @a+Dptr
                                inc Dptr
                                  cjne a,#'@' , jaj212J
                                  clr rs
                MOV SDATA,#80H+40H
                SETB EN
                CLR EN
                LCALL WAIT_LCD
                jmp again
                jaj213e:call write_text
 jmp    Repeat_Data_processingxxn
        jaj212j :cjne a,#'#' , jaj213e
        SETB relaya
clr relayb
        call  waitx
        call  waitx
        call  waitx
        call  waitx
        call  waitx
        call  waitx
        clr relayb
        clr relaya
        call  waitx
        call  waitx
        call  waitx
        call  waitx
        setb rw
SETB relayb
clr relaya
        call waitx
        call waitx
        call waitx
        call waitx
        call waitx
        call waitx
        call waitx
        call waitx
        clr relayb
        clr relaya
        call waitx
        call waitx
        SETB relayb
clr relaya
        call waitx
        call waitx
        clr relayb
        clr relaya
```

```
        call waitx
SETB relayb
clr relaya
        call waitx
        call waitx
        clr relayb
        clr relaya
        call waitx
        call prompting
    ret

    errorr:

setb rw
                clr en
                setb en
                lcall clear_lcd
                lcall init_lcd
                lcall clear_lcd
                clr rs
                mov sdata,#80h+00h
                setb en
                clr en
                lcall wait_lcd
MOV DPTR , #error
AGAINv:
Repeat_Data_processingxxv:
                                loop212Jv: clr a
                                movc a , @a+Dptr
                                inc Dptr
                                  cjne a,#'@' , jaj212Jv
                                  clr rs
                MOV SDATA,#80H+40H
                SETB EN
                CLR EN
                LCALL WAIT_LCD
                jmp againv
                jaj213ek:call write_text
 jmp    Repeat_Data_processingxxv
        jaj212jv :cjne a,#'#' , jaj213ek

    call prompting

    ret
    puk:
setb rw
                clr en
                setb en
                lcall clear_lcd
                lcall init_lcd
                lcall clear_lcd
```

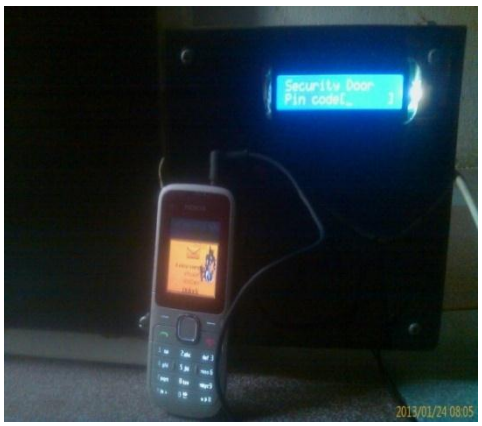
```
                clr rs
                mov sdata,#80h+00h
                setb en
                clr en
                lcall wait_lcd
MOV DPTR , #prompt_PUK
AGAINv1:
Repeat_Data_processingxxv1:
                                loop212Jv1: clr a
                                movc a , @a+Dptr
                                inc Dptr
                                  cjne a,#'@' , jaj212Jv1
                                  clr rs
                MOV SDATA,#80H+40H
                SETB EN
                CLR EN
                LCALL WAIT_LCD
                jmp againv1
                jaj213ek1:call write_text
 jmp     Repeat_Data_processingxxv1
         jaj212jv1 :cjne a,#'#' , jaj213ek1
      ret
     verify2:
     mov  DPTR ,#password_1    ; loading pointer data
     mov r0 ,#Data_Ram_0
Quit_verification1 :
                        loop212q: clr a
                                movc a , @a+Dptr
                                inc Dptr
cjne   @r0 ,#12, Quit_verification1
     inc r0
     inc r7
       cjne  r7, #7  , Quit_verification1
            mov  r7, #00000000b  ; reset counter
     Quit_verification:   mov  DPTR ,#password_2    ; loading pointer data
     mov r0 ,#Data_Ram_0
Repeat_Data_processingxxd :
Quit_verification1d :
                        loop212qd: clr a
                                movc a , @a+Dptr
                                inc Dptr
                                 cjne   @r0,#78,  Quit_verificationda
     inc r0
     inc r7
  cjne  r7, #7  , Quit_verification1d
          mov  r7  ,  #00000000b  ; reset counter
     Quit_verificationda:   inc r6

     cjne  r6 , #3 , error_counter
     mov  r6, #00000000b  ; reset counter
```

```
    error_counter:
    ret
delete_data_process:
mov r0 , #Data_Ram_0
CONTINUE_LOADING : mov @r0 , #' '
inc r0
 cjne r0 , #Data_Ram_9 + 1 , CONTINUE_LOADING
CALL prompting
ret
prompting:
setb rw
                clr en
                setb en
                lcall clear_lcd
                lcall init_lcd
                lcall clear_lcd
                clr rs
                mov sdata,#80h+00h
                setb en
                clr en
                lcall wait_lcd
                                mov Dptr,#prompt
Repeat_Data_processingxx22 :

                                loop21222: clr a
                                movc a , @a+Dptr
                                inc Dptr
                                  cjne a,#'@' , jaj21222f
                                  clr rs
                MOV SDATA,#80H+40H
                SETB EN
                CLR EN
                LCALL WAIT_LCD
                jmp     Repeat_Data_processingxx22
                jaj213zz:
                call write_text
jmp     Repeat_Data_processingxx22
        jaj21222f :cjne a,#'#' , jaj213zz

        mov     a  , #'['
        call write_text
        MOV SDATA,#80H+4fH
                SETB EN
                CLR EN
                LCALL WAIT_LCD
mov     a  , #']'
        call write_text
        MOV SDATA,#80H+49H
                SETB EN
                CLR EN
                LCALL WAIT_LCD
```

```
ret

 ret
wait:;
TT0112:  MOV   R3,#3
     MOV   R2,#208
     MOV   R1,#41
TT1112:  DJNZ  R1,TT1112
     DJNZ  R2,TT1112
     DJNZ  R3,TT1112
     RET
     ogba:
     call wait
  call wait
   call wait
    call wait
    setb rw
                    clr en
                    setb en
                    lcall clear_lcd
                    lcall init_lcd
                    lcall clear_lcd
                    clr rs
                    mov sdata,#80h+00h
                    setb en
                    clr en
                    lcall wait_lcd
     ret
 message1:  db 'GSM enabled DOOR@    lock#'
  message2:  db ' Designed BY @OKWONG , AKAISO#'
  message3:  db 'Mat NO:06/45094#'
password_1 : db '198526#'
password_2 : db '196310#'
access: db ' Access Granted@ Door Open#'
error: db ' Access Denied@ Invalid code#'
prompt: db 'Security Door@Pin code#'
prompt_PUK: db 'Enter PUK pin#'
unlock: db '**Unlock system*@***************#'
PUK_number: db  '38893981#'
end
```

APPENDIX B:

PROJECT GALLERY



Exterior view of controlling unit

Interior view of the controlling unit

Side view of the whole system

Front view of the whole system

Door sliding to show the motor with circuitry

Sliding door in closed position