DEPARTMENT OF Electronic & Computer Engineering

Course(s): All
Level: 1
Module Code: EE1616
Module Name: Level 1 Electronic Workshop
Title of Assessment: PIC Project

School of Engineering & Design
Electronic and Computer Engineering

PIC Project

Final Technical Report

Group Number: Five

Group Members:

Contents

1.The preface

The task of home security system using PIC micro is a task which practice of engineering to solve problems by creating and designing solutions. This task is not just a test of the code programming learning results of PIC16F877A, but also a test of group cooperation ability, team members task assignment, problem design and solution, critical thinking, and other abilities. Of course, it is also quite important for PIC16F877A's coding ability as a basic knowledge. As those who have never officially coded the PIC16F877A, this will undoubtedly be a huge challenge, but the learning ability obtained after final completing the project and the practical knowledge of the PIC during the process is undoubtedly worth it.

In this final report, the task assignments of the group, think of the task, the detailed interpretation of all the code and parts (their final performance in the PC presentation) are detailed. The final discussion will be the summary of the defects and improvement objectives. We are very sorry to say that we have only completed the basic indicators and a small part of the expansion. We have delayed one to two weeks of practice because of the COVID-19 pandemic. We wrote down the direction of improvement at the end of this report.

2. Task objectives and ideas

2.1 Ideas and thinking formulation

According to the aims we have learned during the classes could understand the diversity, functionality, practicality, and security of the current home security system.

By conducting an analysis of the requirements of the task and currently available resources, we developed preliminary group objectives. Despite the limited function of PIC16F877A, but already enough to achieve the following basic functions.

1. System configurations with password protection.

2. Setting security zones.

3. Choosing the active and inactive zones which have chosen in function 2.

4.The buzzer start work when the security zones are invaded.

5. Using password to disarm the alarm.

From above functions, now although the PIC16F877A could do far less than current home security systems in the market, but it is not hard to know that it has the prototype of a home security systems, and already has the most important feature of the home security systems-security.

After this group discussion, a clearly coding direction was developed through the analysis and understanding of the function, every group member started to work from this step. We have made a flow chart to make sure everyone knows their own job clearly. (Details in page 11).

2.2 The actual design

Through the specific discussion and analysis, we have mentioned about the task in the 2.1, we have obtained the goal and research direction of this task. Now let us introduce to you about the **specific function** of this machine. Or it is better to call it a **product description.**

1. Enter the initial password

When you first open the home security system, the system has a initial password ‘1234’. You need enter the correct password to entering the system.

2. Set you own password

When you first enter the system and enter the correct password. The system will force you to set a new password.

3. Show Menu, select

In this interface, you have two choices. If you enter ‘1’, you will set a new password. If you enter ‘2’, you will go to next step and set 4 zones. Users can have different choices to meet their need.

4. Set 4 zones

You can set 4 zones which you want to monitor. Don't worry about entering two identical numbers. Because we have set it, if two identical numbers are input, the LCD will not display.

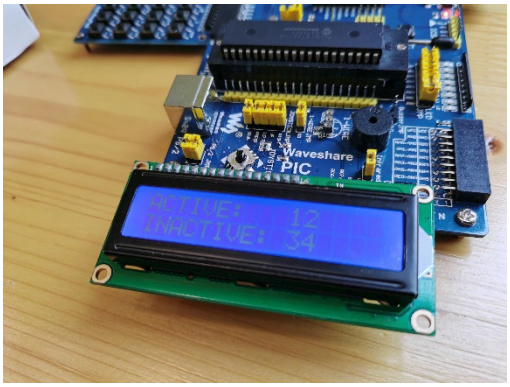
5. Select whether to re-enter

The user may have entered the wrong four-digit area in the previous step. This interface gives them a chance to devise their set. If users enter '1', it means they do not enter wrong, then the system can go to next step. But if users enter '2', it means users want to go back to the previous step and reset the zones.

6. Set security zones or active zones

If users enter ‘1’, it means they want to set this zone to active zone. If users enter ‘0’, it means they don’t want to set this zone to active zone. If the user presses a key other than 0 and 1, it is invalid and will not be displayed on the LCD.

7. Display the active and inactive zones



After the users set the active zones, the LCD will display the active and inactive zones.

8. Detection status

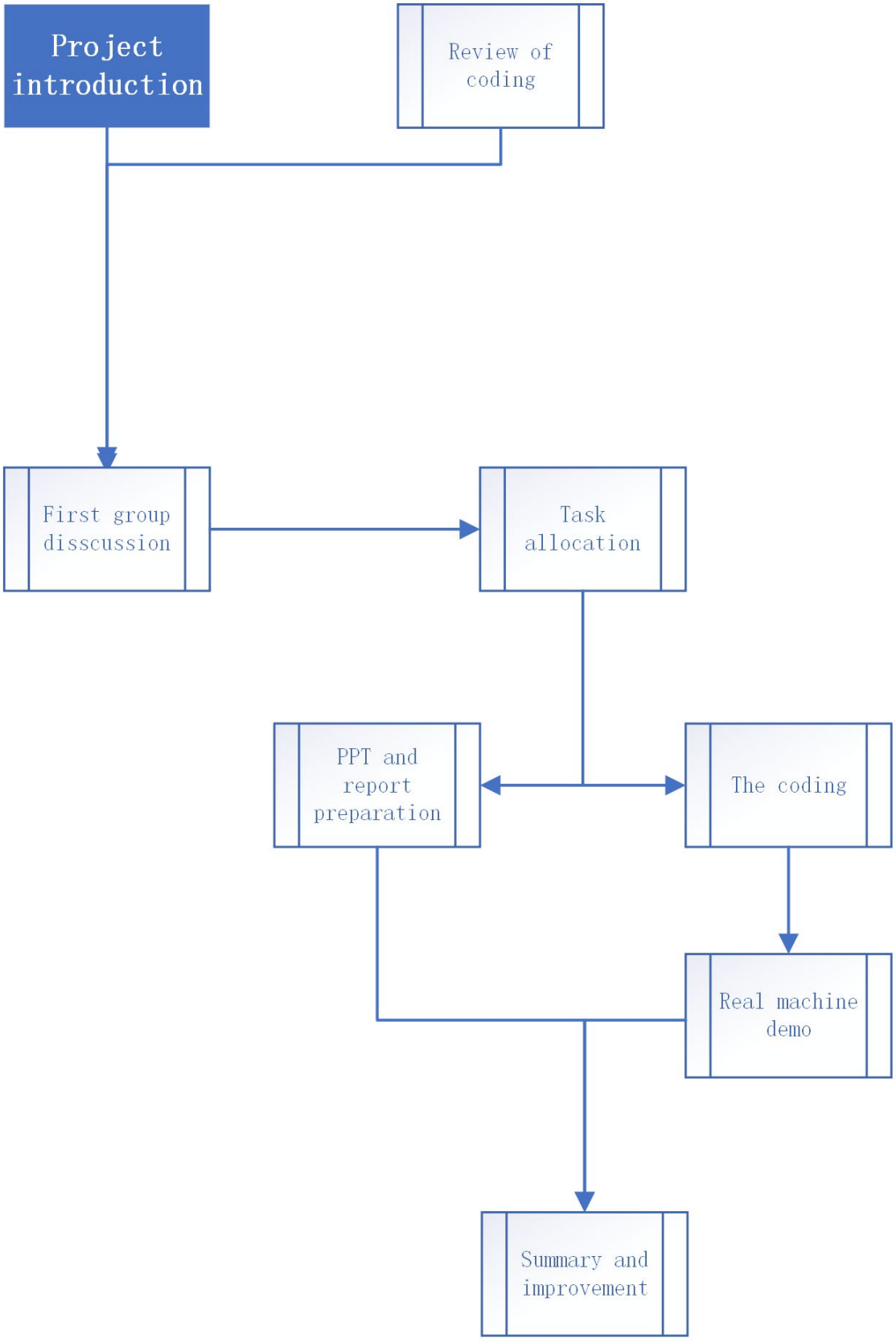
The system enters the detection state. If users enter the active zones the buzzer will alarm and go to next step. If users enter another key, the system will not react.

9. Display the activated area triggered

After trigger the alarm, LCD will display the key which users enter.

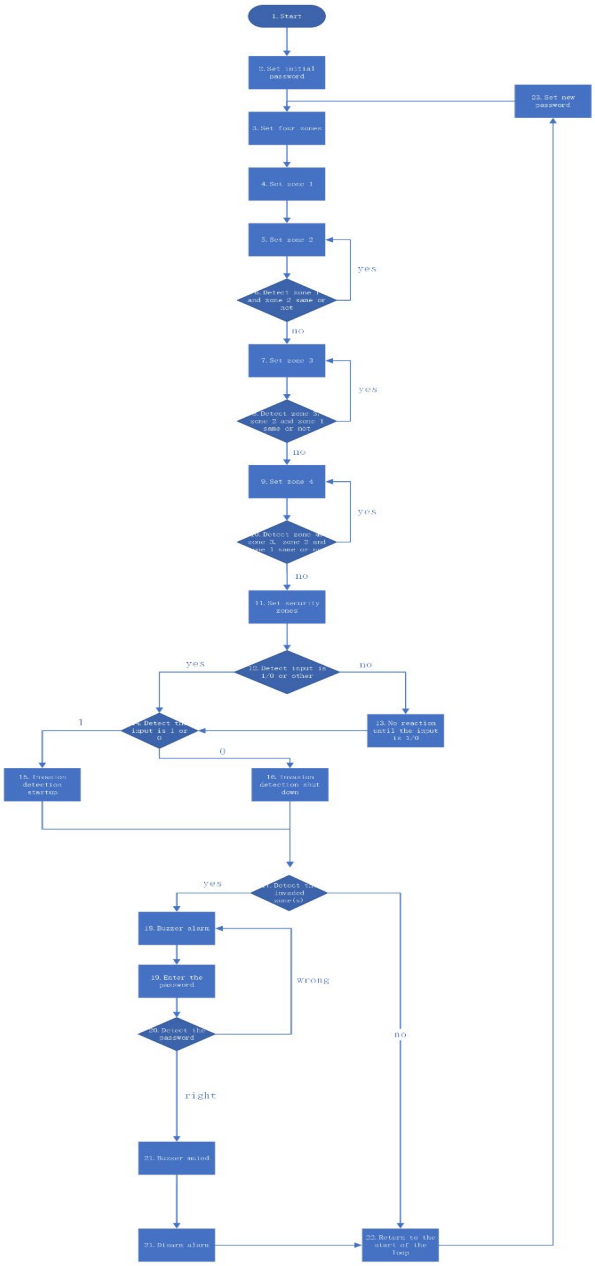
10. Enter correct password to disarm the alarm

If you enter the correct password, the buzzer stops working and the alarm is disarmed. Then the system will go to the initial status. Users need enter password which they set by themselves.


Project introduction
Review of coding
First group disscussion
Task allocation
PPT and report preparation
The coding
Real machine demo
Summary and improvement

3. Code programming with detailed ideas and the code annotation

3.1 Code programming with detailed ideas



This step just as important as “brain” for human, is a crucial step in this task. Compared it with train of thought, it is more like a refinement of the function. It is hard to understand others thought, so I made a flow chart for interpretation.

Step 1 is easy to understand, it’s the start of the whole loop.

The first thing when you get into a security system is to set the password, that’s what step 2 done here.

After set the security password, it is time to set four zones that you want to detect.

Here is question coming, what will happen if you input two or more same zones? So we need to detect the input here, if the input are four different zones, the program will go ahead to next step. But if you input two or more same zones, the same-zone of the after-type input will not be recorded into the register. Until the input is different from the zones you have input before the program will go to next step.

In step 11, you can set the zones you want to make it work, for those four zones, you can input 1 to activate the zones you want and input 0 to shut down the zones that you don’t want. It’s also easy to understand. But what will happen if you input other number like 3,4,5…. ? Actually, nothing will

happen if you input any number besides 1 and 0. Program will go to next step until you have chosen four zones work or not.

From step 17 to step 22 are a unit. For example, if the active zone is zone 1, we used button 1 to be pressed down to simulate the zone was invaded(if the other button was pressed, there will not any reaction), at the same time the buzzer will ring immediately and the LCD pad will display “zone 1 was invaded!” (The same goes for other zones being invaded).

If you want to disarm the alarm, you must input the security password that you have set, if the wrong password you input, the buzzer will continue to alarm. On opposite, if you input the right password, congratulation to you! Yours home security system have successfully defined an invaded.

After the whole loop, the program will return to the step 2, and before step 2, you should input a new security password to start a new loop.

3.2 Code interpretation

3.2.1 Difficult in code

1. **LCD switching display**

   At the initial stage, we are confused about how to change the LCD display. If we complete the screen, we want to know how to change other one.

2. **Monitoring whether there is regional duplication**

   During set 4 zones, if the user accidentally presses two identical keys. We don't want to create two identical regions.

3. **Set whether the area is active**

   Users can enter ‘0’ or ‘1’ to determine whether the zone is security zone. But if users enter other keys, we don’t want store the data and display it on LCD.

4. **Buzzer**

   Triggering the buzzer is a very difficult problem. We haven’t learned in previous lectures.

5. **Debounce**

   KEYPAD often gets stuck. We need to find a way to make our KEYPAD press more smoothly.

3.2.2 Problem solving

1. LCD switching display

```
INITTEST
    CALL        LCDSET
    MOVLW       0X80
    CALL        LCDCMD
    CALL        LCDMSG10

    MOVLW       0XC0
    CALL        LCDCMD
    CALL        GETKEY
    MOVWF       INIT1
    CALL        LCDMSGK
```

- **“LCDSET”**

  It means LCD initialization. Firstly, it can set 8 operation and 2 lines. Secondly, it can set display on/off, cursor on/off, blink on/off. In my code, the program do not need cursor and blink, so I set it off. Thirdly is the most important function, it can clear the previous display. And in the subroutine, it has a “PAUSIT” subroutine, its function is to pause here for 5 seconds, give us time to watch the information on LCD.

- **“LCDCMD”**

  It means configure LCD. For example, if you “MOVLW 0X80” to W register, then you call the subroutine “LCDCMD”, it can set LCD display on the first line. If you “MOVLW 0X80” to W register, then you call the subroutine “LCDCMD”, it can set LCD display on the second line.

- **“LCDMSGK”**

It means display the contents stored in the current W register on the LCD. After you “CALL GETKEY”, then you use this subroutine, it can display the word which you input on KEYPAD.

2. Monitoring whether there is regional duplication

```
DETZONE
    CALL        GETKEY
    MOVWF       STORE
    SUBWF       ZONE1, W
    BTFSC       STATUS, Z
    GOTO        DETZONE
```

```
    MOVF        STORE, W
    SUBWF       ZONE2, W
    BTFSC       STATUS, Z
    GOTO        DETZONE

    MOVF        STORE, W
    SUBWF       ZONE3, W
    BTFSC       STATUS, Z
    GOTO        DETZONE

    MOVF        STORE, W
    SUBWF       ZONE4, W
    BTFSC       STATUS, Z
    GOTO        DETZONE
    RETURN
```

- **“DETZONE”**

This subroutine detects whether the input areas are duplication. “CALL GETKEY” to get a word which you enter. Because the program need this data many times, so I put the data into "STORE”, this register is equivalent to a temporary register. Subtract the data in “ZONE 1” from W. If the two data is equal, the result is “00000000”, so in “STATUS” register, the Z-th bit is one. “BTFSC” means if the value of the bit is 0, skip next instruction. Thus, in this subroutine, if the ZONE 1 is not equal to W, it will “GOTO DETZONE”. Else, execute the next instruction “MOVF STORE,W”.

3. Set whether the area is active

```
;Judge whether the activation area setting is correct
SECURITY
    CALL        GETKEY
    MOVWF       STORE
    XORLW       '0'
    BTFSC       STATUS,Z
    RETURN

    MOVF        STORE,W
    XORLW       '1'
    BTFSC       STATUS,Z
    RETURN
GOTO SECURITY
```

- **“SECURITY”**

  This is very similar to the previous code, call a subroutine “SECURITY”. In the next operation, if users enter the number 1 or 0, the subroutine will return to the main program, else it will loop until it detects 1 or 0.

4. Buzzer

```
BUZZERON
    MOVLW B'11111111'
    MOVWF PORTB
    CALL DELAY
    CLRF PORTB
    CALL DELAY
GOTO BUZZERON

DELAY MOVLW 255
    MOVWF COUNT
DELAY1    DECFSZ COUNT
    GOTO DELAY1
    RETURN
```

This is the initial code design. After search on the schematic diagram, there is a found that the passive buzzer is connect to PORTB1. Passive buzzer can only be triggered in the form of square wave, this means that high and low levels need to be input continuously.

According to this principle, we have completed this series of codes. If the code is run in the file of a new project, the result is very successful, and buzzer is very loud. But if I put these codes into my main program and regard them as a subprogram, the buzzer cannot be triggered. The more serious problem is that we don't know how to stop the triggered buzzer.

Thus, finally, we gave up the code.

```
;Turn on and turn off the buzzer
BUZZERON
    BSF         PORTB, 0
    RETURN

BUZZEROFF
    BCF         PORTB, 0
    RETURN
```

Thus, we got another buzzer. An active buzzer. The active buzzer is very simple to operate, like this figure. We connect I/0 port on the buzzer connect to POTRB,0 port on the PIC board. If you want the buzzer to trigger, you only need to set it to the high level. If you want the buzzer to stop, you only need to set it to the low level.

5. Debounce

```
DEBOUNCE                                    ;A push button tends to create
    MOVLW       0xFF;H'FF'              ;unsettling signal when being pressed
    MOVWF       LOOP1           ;or released. Therefore, a small delay
D_LOOP  DECFSZ  LOOP1           ;routine is created to bypass this
    GOTO        D_LOOP          ;unsettling signal.
RETURN
```

This code is related to the subroutine “GETKEY”, the function of the code is to eliminate switch jitter. At first, the KEYPAD is often stuck at a certain position, and no response is given no matter what key is pressed. At first, we thought it was a keyboard problem, but after many keyboards were replaced, the problem still remained unsolved. Thus, we concentrate on our code. After a long period of debugging, we found that out debounce time is too short, so we extend some time for debounce, the problem is solved.

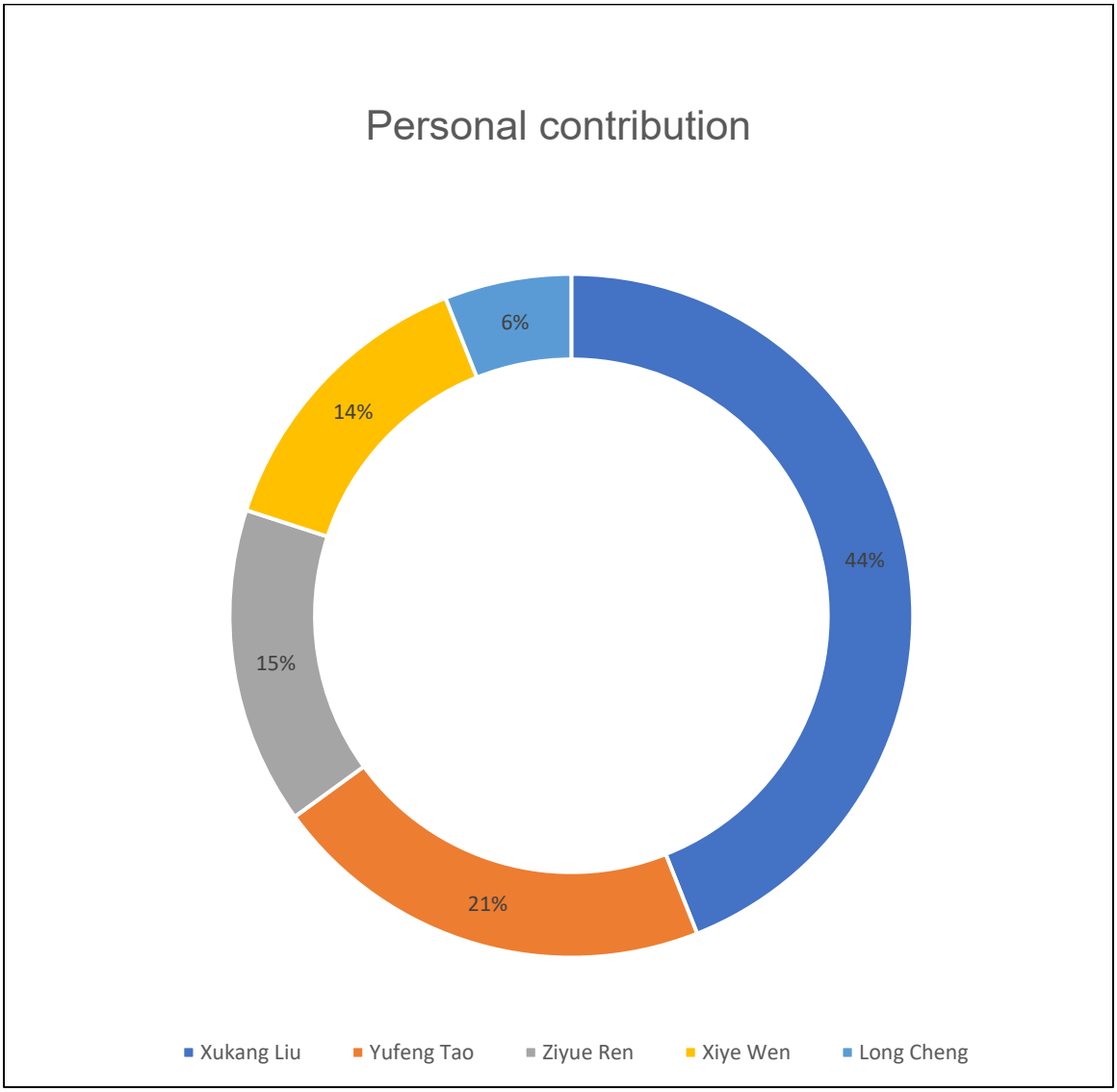
4. Task assignment for the group members

4.1Task assignment

Name and position	Brunel ID	Task(s)
Xukang Liu **leader**	2161047	Assign Tasks Most codes PPT template production and code section Final Report Code Section Presentation
Yufeng Tao **member**	2161037	Provide assembly language related information Summary and summary of weekly tasks Summary of the final report

		Presentation
Ziyue Ren **member**	2161039	Writing of finial report Small part of coding Production of partial PPT Presentation
Xiye Wen **member**	2161043	Menu code writing Area troubleshooting Code rewriting Partial PPT Presentation
Long Cheng **member**	2053536	Partial PPT Presentation

4.2 Personal contribution proportion of the group task



PS: This chart is for reference only, each team member had made their greatest contribution to the group tasks according to their own strengths.

The reference of the calculation:

Degree of participant > Code programming > Ideas of the task > PPT and report made > others.

5. Summary and improvement

In the first week, when we learned about the project content, we first had a preliminary understanding and discussion of our tasks, such as the specific operation steps of compiling, grouping projects, and final presentation. So, in the first week, we did a simple division of labor and wrote a draft of how to implement the functions required by the task. At the same time, since we haven't compiled the PIC for a long time, we have forgotten a lot about assembly, which requires us to take the time to review and revisit. In the second week, through the teacher's further explanation of the project, we have a deeper understanding of the task and a clear goal and plan for the project. By drawing a flowchart, we have a rough framework and steps for the task. In the third week, a more detailed plan was developed than the previous week, along with a clear division of specific tasks for individuals. In the fourth week, each panelist participated in an in-person meeting and summarized their tasks for the previous week. But during the general presentation in the fourth week, we found that the larger program BUG required detailed modifications to the results of the previous week, which took us a lot of time. By the fifth week, our team had completed all the tasks of the project and was able to run the program in its entirety while recording the process using video recording. During the week, we made and revised the PPT and the final report, allowing everyone to participate in the final presentation.

In the one-month task, we found many problems of ourselves, such as unfamiliarity with PIC at the beginning, and did not know how to carry out the specific team division of labor; When a big problem is found halfway

through the task, the team is panicked and doesn't know how to solve it; Toward the end of a task, team information can't be accurately summarized. However, we solved these problems through teamwork, and at the same time, we also understood the importance of teamwork and the importance of each member of the team. Teamwork can exercise the comprehensive ability and comprehensive quality of each member, improve the ability to think about solutions and practical ability when encountering difficulties. Through continuous access to materials and learning, team members understand that they still need to continue to learn in the future and continuously improve their theoretical literacy. In the team practice, each member truly feels the importance of practice, only by putting theory into practice can the truth of theory be tested, and only by letting the code run can we find the problems.

Appendix 1

Assembly code

```
; __config 0xFF32
__CONFIG _FOSC_HS & _WDTE_OFF & _PWRTE_ON & _BOREN_OFF
& _LVP_OFF & _CPD_OFF & _WRT_OFF & _CP_OFF


RES_VECT  CODE    0x0000            ; processor reset vector
LIST    P=16F877
INCLUDE      "P16F877.INC"

; LCD Parameters
LOOP  EQU    H'20'              ;loop counter 1 - general
LOOPA        EQU    H'21'              ;loop counter 2 - LCD use only
CLKCNT       EQU    H'28'              ;125 secs counter 22
STORE        EQU    H'23'              ;general store
RSLINE       EQU    H'24'              ;bit 4 RS line flag for LCD
KEY    EQU    H'25'              ;Count of keys
```

```
LOOP1          EQU    H'26'

STORE1  EQU     H'27'       ;The first password
STORE2  EQU     H'28'       ;The second password
STORE3  EQU     H'29'       ;The third password
STORE4  EQU     H'30'       ;The fourth password

ZONE1   EQU     H'31'       ;The 1st monitoring zone
ZONE2   EQU     H'32'       ;2nd
ZONE3   EQU     H'33'       ;3th
ZONE4   EQU     H'34'       ;The 4th monitoring zones

SECUR1  EQU     H'35'       ;Set active and inactive zones
SECUR2  EQU     H'36'
SECUR3  EQU     H'37'
SECUR4  EQU     H'38'

DETECTX EQU     H'39'
TRIGGER EQU     H'40'

WORD1   EQU     H'42'
WORD2   EQU     H'43'
WORD3   EQU     H'44'
WORD4   EQU     H'45'

INIT1   EQU     H'46'
INIT2   EQU     H'47'
INIT3   EQU     H'48'
INIT4   EQU     H'49'

ORG             0x00
```

```
GOTO START

; ----Main program starts Here----
START
   BANKSEL  PORTB
   CLRF        PORTB
   CLRF       SECUR1
   CLRF       ZONE2
   CLRF       ZONE3
   CLRF       ZONE4
   BANKSEL  TRISB
   CLRF        TRISD          ;Port D0-D7 as output
   CLRF       TRISB
   MOVLW    B'00000110'    ;set timer ratio 1:128
   MOVWF    OPTION_REG
   CLRF        TRISC

   BANKSEL  PORTB
MAIN
   GOTO SUBMAIN

TABLCD       ADDWF       PCL,F            ;LCD initialisation table
   RETLW     B'00111000'   ;Function Set: 8-bit operation and 2 lines
   RETLW     B'00001100'   ;Display ON/OFF Control: set display on, cursor off, blink
off
   RETLW     B'00000001'   ;Clear Display
   RETLW     B'00000010'   ;Return Home: Set cursor to home position
   RETLW     0X00          ;End of initialisation table

MESSAG       ADDWF       PCL,F
   DT "SET PASSWORD:",0 ;0 is an indicator for end of string
```

```
MESSAG1 ADDWF  PCL,F
   DT"SET 4 Zones:",0

MESSAG2 ADDWF  PCL,F
   DT"Set Active zone",0

MESSAG3 ADDWF  PCL,F
   DT"1:ACTIVE 0:INACT ",0

MESSAG4 ADDWF  PCL,F
   DT": ",0

MESSAG5 ADDWF  PCL,F
   DT"ZONE ",0

MESSAG6 ADDWF  PCL,F
   DT"ACTIVE:   ",0

MESSAG7 ADDWF  PCL,F
   DT"***DETECTING***",0

MESSAG8 ADDWF  PCL,F
   DT"INACTIVE: ",0

MESSAG9 ADDWF  PCL,F
   DT"Trigger Zone:",0

MESSAG10 ADDWF PCL,F
   DT"ENTER PASSWORD",0
```

```
MESSAG11 ADDWF PCL,F
   DT"1:SET NEW KEY",0

MESSAG12 ADDWF PCL,F
   DT"2:SET AREAS",0

MESSAG13 ADDWF PCL,F
   DT"Continue?:",0

MESSAG14 ADDWF PCL,F
   DT"1:YES  2:NO",0


;sub-routine to send a string of LCD command to LCD module
LCDSET      CALL  PAUSIT
   CLRF     LOOP          ;clr LCD set-up loop
   CLRF     RSLINE              ;clear RS line for instruction send
LCDST       MOVF LOOP,W               ;get table address
   CALL     TABLCD              ;get set-up instruction
   XORLW    0X00          ;0x00 is the indicator for last LCD instruction
   BTFSC    STATUS,Z      ;has last LCD set-up instruction now been done?
   GOTO     SETEND              ;YES, so end the LCD SET routine
   CALL     LCDOUT              ;No, send the instruction to LCD
   INCF     LOOP,F              ;inc loop
   GOTO     LCDST               ;get the next set-up instruction
   SETEND   CALL  PAUSIT                ;perform second 1/5th sec delay
   RETURN                       ;to allow final LCD command to occur

;sub-routine to send a string of alphanumeric letter to LCD module
LCDMSG
   CLRF     LOOP          ;clear loop
```

```
   BSF         RSLINE,4      ;set RS for data send
LCDMS       MOVF LOOP,W              ;get table address
   CALL        MESSAG             ;get message letter
   XORLW     0X00           ;0x00 is the indicator for last data
   BTFSC       STATUS,Z     ;has last LCD letter been sent?
   GOTO        MSGEND             ;YES, so end the DATA SEND routine
   CALL        LCDOUT             ;No, send the data to LCD for display
   INCF        LOOP,F             ;inc loop
   GOTO        LCDMS              ;repeat for next one letter
MSGEND    RETURN


;Set 4 zones monitoring
LCDMSG1
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS1  MOVF  LOOP,W
   CALL        MESSAG1
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND1
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS1
   MSGEND1  RETURN

LCDMSG2
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS2  MOVF  LOOP,W
   CALL        MESSAG2
```

```
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND2
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS2
   MSGEND2 RETURN

LCDMSG3
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS3  MOVF  LOOP,W
   CALL        MESSAG3
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND3
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS3
   MSGEND3 RETURN

LCDMSG4
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS4  MOVF  LOOP,W
   CALL        MESSAG4
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND4
   CALL        LCDOUT
   INCF        LOOP,F
```

```
   GOTO        LCDMS4
   MSGEND4 RETURN

LCDMSG5
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS5  MOVF  LOOP,W
   CALL        MESSAG5
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND5
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS5
   MSGEND5 RETURN

LCDMSG6
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS6  MOVF  LOOP,W
   CALL        MESSAG6
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND6
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS6
   MSGEND6 RETURN

LCDMSG7
   CLRF        LOOP
```

```
   BSF         RSLINE,4
   LCDMS7  MOVF  LOOP,W
   CALL        MESSAG7
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND7
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS7
   MSGEND7 RETURN

LCDMSG8
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS8  MOVF  LOOP,W
   CALL        MESSAG8
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND8
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS8
   MSGEND8 RETURN

LCDMSG9
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS9  MOVF  LOOP,W
   CALL        MESSAG9
   XORLW       0X00
   BTFSC       STATUS,Z
```

```
   GOTO        MSGEND9
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS9
   MSGEND9 RETURN

LCDMSG10
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS10  MOVF  LOOP,W
   CALL        MESSAG10
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND10
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS10
   MSGEND10 RETURN

LCDMSG11
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS11  MOVF  LOOP,W
   CALL        MESSAG11
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND11
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS11
   MSGEND11 RETURN
```

```
LCDMSG12
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS12  MOVF  LOOP,W
   CALL        MESSAG12
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND12
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS12
   MSGEND12 RETURN

LCDMSG13
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS13  MOVF  LOOP,W
   CALL        MESSAG13
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND13
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS13
   MSGEND13 RETURN

LCDMSG14
   CLRF        LOOP
   BSF         RSLINE,4
   LCDMS14  MOVF  LOOP,W
```

```
   CALL        MESSAG14
   XORLW       0X00
   BTFSC       STATUS,Z
   GOTO        MSGEND14
   CALL        LCDOUT
   INCF        LOOP,F
   GOTO        LCDMS14
   MSGEND14 RETURN


;sub-routine to send a string of alphanumeric letter to LCD module
LCDMSGK
   BSF         RSLINE,4      ;set RS for data send
   CALL        LCDOUT              ;send the data to LCD for display
   RETURN


;sub-routine to send one byte data to LCD (which can be a command or alphanumeric
letter
LCDOUT
   MOVWF    STORE              ;temp store data
   MOVLW    D'250';D'250'      ;set minimum time between sending full bytes to
   MOVWF    LOOPA              ;LCD - value of 250 seems OK for this prog with
DELAY       DECFSZ       LOOPA,F            ;XTAL clk of upto 20MHz
   GOTO        DELAY              ;keep decrementing LOOPA until zero
   CALL        SENDIT             ;send data
   RETURN

;used by LCDOUT to send a 8-bit of one byte data to LCD
SENDIT
   MOVF        STORE,W            ;get data byte
```

```
   BCF PORTC,4
   BTFSC RSLINE,4   ; RSLINE 1-data 0-instraction
   BSF PORTC,4
   MOVWF    PORTD              ;output the byte
   BSF      PORTC,6            ;set E line high
   BCF      PORTC,6            ;set E line low
   RETURN


;sub-routine to send an alphanumeric letter in W register to LCD module
LCDDATA    BSF          RSLINE,4      ;set RS=1. This will set the LCD to Data
mode
   CALL     LCDOUT
   RETURN


;sub-routine to send a LCD command in W register to LCD module
LCDCMD     CLRF  RSLINE               ;set RS=0. This will set the LCD to
Command mode
   CALL     LCDOUT
   RETURN


;delay subroutine
PAUSIT     MOVLW     30                    ;set delay counter to 30
   MOVWF    CLKCNT          ;(for 1/150th sec x 30)
   CLRF     INTCON          ;clear interupt flag
PAUSE      BTFSSINTCON,2     ;initial 1/5th sec wait before setting up LCD has a
timer time-out been detected?
   GOTO     PAUSE           ;no
   BCF      INTCON,2     ;yes
```

```
   DECFSZ    CLKCNT,F    ;dec counter, is it zero?
   GOTO      PAUSE             ;no
   RETURN                      ;yes


 GETKEY
   MOVLW     H'FF'
   MOVWF     PORTD


   BCF       PORTD,0           ;Check ROW1
   CALL      KEY_ROW1  ;Check if there is any keypress in Column A,B,C or D of
ROW1 and return a value for a corresponding key.
                               ;If no keypress found, the subroutine will return 0xFF in W
regiter.
   MOVWF     KEY               ;Store the return value from W to KEY (a variable which
key the return keypress value).
   XORLW     0XFF              ;Check if value in W is equal to 0xFF.
   BTFSS     STATUS,Z    ;Yes. This means the KEY_ROW1 has returned a 0xFF in
the W; therefore no keypress detected.
   GOTO      FOUND1            ;No. A keypress is found from


   BSF       PORTD,0           ;Check ROW2
   BCF       PORTD,1           ;
   CALL      KEY_ROW2  ;Check if there is any keypress in Column A,B,C or D of
ROW2 and return a value for a corresponding key.
                               ;If no keypress found, the subroutine will return 0xFF in W
regiter.
```

```
   MOVWF    KEY          ;Store the return value from W to KEY (a variable which
key the return keypress value).
   XORLW    0XFF         ;Check if value in W is equal to 0xFF.
   BTFSS    STATUS,Z     ;Yes. This means the KEY_ROW2 has returned a 0xFF in
the W; therefore no keypress detected.
   GOTO     FOUND2            ;No. A keypress is found from



   BSF      PORTD,1           ;Check ROW3
   BCF      PORTD,2

   CALL     KEY_ROW3  ;Check if there is any keypress in Column A,B,C or D of
ROW3 and return a value for a corresponding key.
                      ;If no keypress found, the subroutine will return 0xFF in W
regiter.
   MOVWF    KEY          ;Store the return value from W to KEY (a variable which
key the return keypress value)
   XORLW    0XFF         ;Check if value in W is equal to 0xFF.
   BTFSS    STATUS,Z     ;Yes. This means the KEY_ROW3 has returned a 0xFF in
the W; therefore no keypress detected.
   GOTO     FOUND3            ;No. A keypress is found from



   BSF      PORTD,2           ;Check ROW4
   BCF      PORTD,3
   CALL     KEY_ROW4  ;Check if there is any keypress in Column A,B,C or D of
ROW4 and return a value for a corresponding key.
                      ;If no keypress found, the subroutine will return 0xFF in W
regiter.
```

```
   MOVWF    KEY          ;Store the return value from W to KEY (a variable which
key the return keypress value)
   XORLW    0XFF         ;Check if value in W is equal to 0xFF.
   BTFSS    STATUS,Z     ;Yes. This means the KEY_ROW3 has returned a 0xFF in
the W; therefore no keypress detected.
   GOTO     FOUND4            ;No. A keypress is found from



   GOTO     GETKEY            ;Repeat the scanning non-stop


FOUND1
   CALL     KEY_ROW1  ;Check whether if any key in ROW1 still being pressed. If
no key is pressed, a 0xFF will be returned in W.
   XORLW    H'FF'        ;Check if W is equal to 0xFF
   BTFSS    STATUS,Z     ;Yes. This means the keypress is released by user.
   GOTO     FOUND1            ;No. This means the keypress is still hold by user.
Goto FOUND1 and check again until user release the keypress.
   MOVFW    KEY               ;Proceed when user release the keypress and move
the KEY value into W.
   RETURN                     ;Return to the main program with the detected
keypress value (KEY) in W.


FOUND2
   CALL     KEY_ROW2  ;Check whether if any key in ROW2 still being pressed. If
no key is pressed, a 0xFF will be returned in W.
   XORLW    H'FF'        ;Check if W is equal to 0xFF
   BTFSS    STATUS,Z     ;Yes. This means the keypress is released by user.
```

```
   GOTO     FOUND2          ;No. This means the keypress is still hold by user.
Goto FOUND1 and check again until user release the keypress.
   MOVFW    KEY             ;Proceed when user release the keypress and move
the KEY value into W.
   RETURN                   ;Return to the main program with the detected
keypress value (KEY) in W.


FOUND3
   CALL     KEY_ROW3  ;Check whether if any key in ROW3 still being pressed. If
no key is pressed, a 0xFF will be returned in W.
   XORLW    H'FF'     ;Check if W is equal to 0xFF
   BTFSS    STATUS,Z  ;Yes. This means the keypress is released by user.
   GOTO     FOUND3          ;No. This means the keypress is still hold by user.
Goto FOUND1 and check again until user release the keypress.
   MOVFW    KEY             ;Proceed when user release the keypress and move
the KEY value into W.
   RETURN                   ;Return to the main program with the detected
keypress value (KEY) in W.


FOUND4
   CALL     KEY_ROW4  ;Check whether if any key in ROW4 still being pressed. If
no key is pressed, a 0xFF will be returned in W.
   XORLW    H'FF'     ;Check if W is equal to 0xFF
   BTFSS    STATUS,Z  ;Yes. This means the keypress is released by user.
   GOTO     FOUND4          ;No. This means the keypress is still hold by user.
Goto FOUND1 and check again until user release the keypress.
   MOVFW    KEY             ;Proceed when user release the keypress and move
the KEY value into W.
```

```
   RETURN                  ;Return to the main program with the detected
keypress value (KEY) in W.

;check the "row" of the Matrix Keypad
KEY_ROW1
   CALL     DEBOUNCE
   BTFSS    PORTD,7         ;Check Column A of ROW1
   RETLW    '1'             ;Return an ASCII for character '1' that is 0x31
   BTFSS    PORTD,6         ;Check Column B of ROW1
   RETLW    '2'             ;Return an ASCII for character '2' that is 0x32
   BTFSS    PORTD,5         ;Check Column C of ROW1
   RETLW    '3'             ;Return an ASCII for character '3' that is 0x33
   BTFSS    PORTD,4         ;Check Column D of ROW1
   RETLW    'S'             ;Return an ASCII for character 'S(STOP)' that is
0x53
   RETLW    H'FF'

KEY_ROW2
   CALL     DEBOUNCE
   BTFSS    PORTD,7         ;Check Column A of ROW2
   RETLW    '4'             ;Return an ASCII for character '4' that is 0x34
   BTFSS    PORTD,6         ;Check Column B of ROW2
   RETLW    '5'             ;Return an ASCII for character '5' that is 0x35
   BTFSS    PORTD,5         ;Check Column C of ROW2
   RETLW    '6'             ;Return an ASCII for character '6' that is 0x36
   BTFSS    PORTD,4         ;Check Column D of ROW2
   RETLW    'G'             ;Return an ASCII for character 'G(GO)' that is 0x47
   RETLW    H'FF'

KEY_ROW3
   CALL     DEBOUNCE
```

```
   BTFSS    PORTD,7          ;Check Column A of ROW3
   RETLW    '7'              ;Return an ASCII for character '7' that is 0x37
   BTFSS    PORTD,6          ;Check Column B of ROW3
   RETLW    '8'              ;Return an ASCII for character '8' that is 0x38
   BTFSS    PORTD,5          ;Check Column C of ROW3
   RETLW    '9'              ;Return an ASCII for character '9' that is 0x39
   BTFSS    PORTD,4          ;Check Column D of ROW3
   RETLW    'L'              ;Return an ASCII for character 'L(LOCK)' that is
0x4C
   RETLW    H'FF'

KEY_ROW4
   CALL     DEBOUNCE
   BTFSS    PORTD,7          ;Check Column A of ROW4
   RETLW    'E'              ;Return an ASCII for character 'E(ENT)' that is
0x45
   BTFSS    PORTD,6          ;Check Column B of ROW4
   RETLW    '0'              ;Return an ASCII for character '0' that is 0x30
   BTFSS    PORTD,5          ;Check Column C of ROW4
   RETLW    'C'              ;Return an ASCII for character 'C(ESC)' that is
0x43
   BTFSS    PORTD,4          ;Check Column D of ROW4
   RETLW    'P'              ;Return an ASCII for character 'P(POWER)' that is
0x50
   RETLW    H'FF'
  ;----------------
DEBOUNCE                         ;A push button tends to create
   MOVLW    0xFF;H'FF'       ;unsettling signal when being pressed
   MOVWF    LOOP1            ;or released. Therefore, a small delay
D_LOOP   DECFSZ    LOOP1          ;routine is created to bypass this
   GOTO     D_LOOP           ;unsettling signal.
```

```
RETURN

;Monitor whether the set area is repeated
DETZONE
   CALL       GETKEY
   MOVWF      STORE
   SUBWF      ZONE1,W
   BTFSC      STATUS,Z
   GOTO       DETZONE

   MOVF       STORE,W
   SUBWF      ZONE2,W
   BTFSC      STATUS,Z
   GOTO       DETZONE

   MOVF       STORE,W
   SUBWF      ZONE3,W
   BTFSC      STATUS,Z
   GOTO       DETZONE

   MOVF       STORE,W
   SUBWF      ZONE4,W
   BTFSC      STATUS,Z
   GOTO       DETZONE
   RETURN

;Judge whether the activation area setting is correct
SECURITY
   CALL       GETKEY
   MOVWF      STORE
   XORLW      '0'
```

```
   BTFSC      STATUS,Z
   RETURN

   MOVF       STORE,W
   XORLW      '1'
   BTFSC      STATUS,Z
   RETURN
GOTO SECURITY

;Display active zones
ACTIVE
   MOVF       SECUR1,W
   XORLW      '1'
   BTFSC      STATUS,Z
   CALL       DISLCD1

   MOVF       SECUR2,W
   XORLW      '1'
   BTFSC      STATUS,Z
   CALL       DISLCD2

   MOVF       SECUR3,W
   XORLW      '1'
   BTFSC      STATUS,Z
   CALL       DISLCD3

   MOVF       SECUR4,W
   XORLW      '1'
   BTFSC      STATUS,Z
   CALL       DISLCD4
   RETURN
```

```
;Display inactive zones
INACTIVE
   MOVF       SECUR1,W
   XORLW      '0'
   BTFSC      STATUS,Z
   CALL       DISLCD1

   MOVF       SECUR2,W
   XORLW      '0'
   BTFSC      STATUS,Z
   CALL       DISLCD2

   MOVF       SECUR3,W
   XORLW      '0'
   BTFSC      STATUS,Z
   CALL       DISLCD3

   MOVF       SECUR4,W
   XORLW      '0'
   BTFSC      STATUS,Z
   CALL       DISLCD4
   RETURN

DISLCD1
   MOVF       ZONE1,W
   BSF        RSLINE,4
   CALL       LCDOUT
   RETURN

DISLCD2
```

```
   MOVF       ZONE2,W
   BSF        RSLINE,4
   CALL       LCDOUT
   RETURN

DISLCD3
   MOVF       ZONE3,W
   BSF        RSLINE,4
   CALL       LCDOUT
   RETURN

DISLCD4
   MOVF       ZONE4,W
   BSF        RSLINE,4
   CALL       LCDOUT
   RETURN

;Detect whether the pressed key is monitoring areas
DETECT
   CLRF       DETECTX
   CALL       GETKEY
   MOVWF      STORE
   SUBWF      ZONE1,W
   BTFSC      STATUS,Z
   CALL       DETECT1

   MOVF       STORE,W
   SUBWF      ZONE2,W
   BTFSC      STATUS,Z
   CALL       DETECT2
```

```
   MOVF       STORE,W
   SUBWF      ZONE3,W
   BTFSC      STATUS,Z
   CALL       DETECT3

   MOVF       STORE,W
   SUBWF      ZONE4,W
   BTFSC      STATUS,Z
   CALL       DETECT4
   CALL       DETECTA
   RETURN

DETECT1
   MOVF       SECUR1,W
   XORLW      '1'
   BTFSS      STATUS,Z
   RETURN
   CALL       BUZZERON
   MOVLW      '2'
   MOVWF      DETECTX
   RETURN

DETECT2
   MOVF       SECUR2,W
   XORLW      '1'
   BTFSS      STATUS,Z
   RETURN
   CALL       BUZZERON
   MOVLW      '2'
   MOVWF      DETECTX
   RETURN
```

```
DETECT3
   MOVF       SECUR3,W
   XORLW      '1'
   BTFSS      STATUS,Z
   RETURN
   CALL       BUZZERON
   MOVLW      '2'
   MOVWF      DETECTX
   RETURN

DETECT4
   MOVF       SECUR4,W
   XORLW      '1'
   BTFSS      STATUS,Z
   RETURN
   CALL       BUZZERON
   MOVLW      '2'
   MOVWF      DETECTX
   RETURN

DETECTA
   MOVF       DETECTX,W
   XORLW      '2'
   BTFSS      STATUS,Z
   GOTO       DETECT
   RETURN

;Turn on and turn off the buzzer
BUZZERON
   BSF        PORTB,0
```

```
  RETURN

BUZZEROFF
   BCF        PORTB,0
  RETURN

;The user enters the password to disarm the alarm
DISARM
  CALL        LCDSET
  MOVLW       0X80
  CALL        LCDCMD
  CALL        LCDMSG10

  MOVLW       0XC0
  CALL        LCDCMD
  CALL        GETKEY
  MOVWF       WORD1
  CALL        LCDMSGK

  CALL        GETKEY
  MOVWF       WORD2
  CALL        LCDMSGK

  CALL        GETKEY
  MOVWF       WORD3
  CALL        LCDMSGK

  CALL        GETKEY
  MOVWF       WORD4
  CALL        LCDMSGK
```

```
  MOVF        WORD1,W
  SUBWF       STORE1,W
  BTFSS       STATUS,Z
  GOTO        DISARM

  MOVF        WORD2,W
  SUBWF       STORE2,W
  BTFSC       STATUS,Z
  GOTO        DISARM

  MOVF        WORD3,W
  SUBWF       STORE3,W
  BTFSS       STATUS,Z
  GOTO        DISARM

  MOVF        WORD4,W
  SUBWF       STORE4,W
  BTFSS       STATUS,Z
  GOTO        DISARM
  RETURN

;Test the initial password, the default password is "1234"
INITTEST
  CALL        LCDSET
  MOVLW       0X80
  CALL        LCDCMD
  CALL        LCDMSG10

  MOVLW       0XC0
  CALL        LCDCMD
  CALL        GETKEY
```

```
MOVWF       INIT1
CALL        LCDMSGK

CALL        GETKEY
MOVWF       INIT2
CALL        LCDMSGK

CALL        GETKEY
MOVWF       INIT3
CALL        LCDMSGK

CALL        GETKEY
MOVWF       INIT4
CALL        LCDMSGK

MOVF        INIT1,W
XORLW       '1'
BTFSS       STATUS,Z
GOTO        INITTEST

MOVF        INIT2,W
XORLW       '2'
BTFSS       STATUS,Z
GOTO        INITTEST

MOVF        INIT3,W
XORLW       '3'
BTFSS       STATUS,Z
GOTO        INITTEST

MOVF        INIT4,W
```

```
  XORLW       '4'
  BTFSS       STATUS,Z
  GOTO        INITTEST
  RETURN

;Set four new passwords
SETKEY
  CALL      LCDSET
   MOVLW    0X80
   CALL     LCDCMD
   CALL     LCDMSG

   MOVLW    0XC0
   CALL     LCDCMD
   CALL     GETKEY
   MOVWF     STORE1
   CALL      LCDMSGK

   CALL     GETKEY
   MOVWF     STORE2
   CALL     LCDMSGK

   CALL     GETKEY
   MOVWF     STORE3
   CALL     LCDMSGK

   CALL     GETKEY
   MOVWF     STORE4
   CALL     LCDMSGK
   RETURN
```

```
SELECT
   CALL      GETKEY
   MOVWF     STORE
   XORLW     '1'
   BTFSC     STATUS,Z
   GOTO      MAIN1

   MOVF      STORE,W
   XORLW     '2'
   BTFSC     STATUS,Z
   GOTO      MAIN3
   GOTO      SELECT

KEEP
   CALL      LCDSET
   MOVLW      0X80
   CALL      LCDCMD
   CALL      LCDMSG13

   MOVLW      0XC0
   CALL      LCDCMD
   CALL      LCDMSG14

   CALL      GETKEY
   MOVWF      STORE
   XORLW      '1'
   BTFSC      STATUS,Z
   GOTO       MAIN4
   MOVF       STORE,W
   XORLW      '2'
   BTFSC      STATUS,Z
```

```
   GOTO        MAIN3
   GOTO        KEEP

KEEP1
   CALL        LCDSET
   MOVLW       0X80
   CALL        LCDCMD
   CALL        LCDMSG13

   MOVLW       0XC0
   CALL        LCDCMD
   CALL        LCDMSG14

   CALL        GETKEY
   MOVWF       STORE
   XORLW       '1'
   BTFSC       STATUS,Z
   GOTO        MAIN5
   MOVF        STORE,W
   XORLW       '2'
   BTFSC       STATUS,Z
   GOTO        MAIN4
   GOTO        KEEP

;This is the main subroutine
SUBMAIN
  CALL        INITTEST
MAIN1
  CALL        SETKEY
MAIN2
  CALL        DISARM
```

```
  CALL        LCDSET
  MOVLW       0X80
  CALL        LCDCMD
  CALL        LCDMSG11

  MOVLW       0XC0
  CALL        LCDCMD
  CALL        LCDMSG12
  GOTO        SELECT
MAIN3
  CALL        LCDSET              ;Configure the LCD display
  MOVLW     0X80
  CALL      LCDCMD
  CALL      LCDMSG1

  MOVLW     0XC0
  CALL      LCDCMD

  CALL      GETKEY
  MOVWF     ZONE1
  CALL      LCDMSGK

  CALL      DETZONE
  MOVF      STORE,W
  MOVWF     ZONE2
  CALL      LCDMSGK

  CALL      DETZONE
  MOVF      STORE,W
  MOVWF     ZONE3
  CALL      LCDMSGK
```

```
   CALL     DETZONE
   MOVF     STORE,W
   MOVWF    ZONE4
   CALL     LCDMSGK

   GOTO     KEEP
MAIN4
   CALL     LCDSET
   MOVLW    0X80          ;explain the function of 1 and 0
   CALL     LCDCMD
   CALL     LCDMSG2

   MOVLW    0XC0
   CALL     LCDCMD
   CALL     LCDMSG3
   CALL     PAUSIT

   CALL     LCDSET
   MOVLW    0X80          ;Set activation zones
   CALL     LCDCMD
   CALL     LCDMSG5
   MOVF     ZONE1,W
   CALL     LCDMSGK
   CALL     LCDMSG4
   CALL     SECURITY
   MOVF     STORE,W
   MOVWF    SECUR1
   CALL     LCDMSGK

   MOVLW    0XC0
```

```
CALL     LCDCMD
CALL     LCDMSG5
MOVF     ZONE2,W
CALL     LCDMSGK
CALL     LCDMSG4
CALL     SECURITY
MOVF     STORE,W
MOVWF    SECUR2
CALL     LCDMSGK

CALL     LCDSET
MOVLW    0X80
CALL     LCDCMD
CALL     LCDMSG5
MOVF     ZONE3,W
CALL     LCDMSGK
CALL     LCDMSG4
CALL     SECURITY
MOVF     STORE,W
MOVWF    SECUR3
CALL     LCDMSGK

MOVLW    0XC0
CALL     LCDCMD
CALL     LCDMSG5
MOVF     ZONE4,W
CALL     LCDMSGK
CALL     LCDMSG4
CALL     SECURITY
MOVF     STORE,W
MOVWF    SECUR4
```

```
   CALL     LCDMSGK

   CALL     KEEP1
MAIN5
   CALL     LCDSET        ;Display active and inactive zones
   MOVLW    0X80
   CALL     LCDCMD
   CALL     LCDMSG6
   CALL     ACTIVE

   MOVLW    0XC0
   CALL     LCDCMD
   CALL     LCDMSG8
   CALL     INACTIVE

   CALL     LCDSET        ;Detecting active zones
   MOVLW    0X80
   CALL     LCDCMD
   CALL     LCDMSG7
   CALL     DETECT
   MOVF     STORE,W
   MOVWF    TRIGGER

   CALL     LCDSET
   MOVLW    0X80
   CALL     LCDCMD
   CALL     LCDMSG9

   MOVLW    0XC0
   CALL     LCDCMD
   CALL     LCDMSG5
```

```
    MOVF    TRIGGER,W
    CALL    LCDMSGK
    CALL    PAUSIT

    CALL    DISARM          ;Disarm the alarm, turn off the buzzer
    CALL    BUZZEROFF

GOTO MAIN2
END
```

Appendix 2

Process report

EE1616 - Workshop

Group Project Progress and Action Plan Sheet

Note: This form should be completed in electronic format, but a hard copy should be printed out for submission at the end of each PIC Project lab session.

Date: 2022/9/7

Group members present: Xukang Liu, Yufeng Tao, Ziyue Ren, Xiye Wen, Long Chen

1. Progress during this week

In this class, we have a preliminary understanding of our task. The object contained compilation, group project, final presentation and so on. Thus, we made a simple division of labor. Compilation is the most important part of the project, so we temporarily assigned three people to this task. After read "Group project assignment", we discussed what our project should do. Finally, we write a draft to explain how to realize these functions.

2. Problems encountered

Because we haven't programmed for a long time, we have forgotten a lot about assembly. Moreover, there are many functions in this project that look very complex, and we don't have some frames, so it is difficult for us.

3. Objectives for next week

To familiar with assembly language. Obtain PIC broad, make more specific measures and compile it on the computer.

Form Completed by: Xukang Liu

Group Project Progress and Action Plan Sheet

Note: This form should be completed in electronic format but a hard copy should be printed out for submission at the end of each PIC Project lab session.

Date:2022/9/14

Group members present: Xukang Liu

Yufeng Tao

Ziyue Ren

Xiye Wen

Long Chen

1. Progress during this week

In this lesson, the instructor's further explanation of the project gave us a deeper understanding of our tasks. There is a clear direction and plan for the project. 1. Enter the password to enter the system settings 2. Set 4 safe areas, for example, button 1 on the keypad indicates room1 3. Set the monitoring status of 4 safe areas, for example set room1 to active 4. When pressing button 1 is detected to be dangerous, buzzer alarm 5. When entering another password to dismiss the alarm.

2. Problems encountered

There is no definitive workaround for how activation or detection of security zones needs to be implemented.

It is not clear how to implement the specific programming of the buzzer alarm.

Due to the epidemic situation, the group members cannot communicate offline, some specific operations cannot be implemented more effectively, and the division of labor for each person is not particularly clear.

3. Objectives for next week

Check the information and ask the tutor to solve the specific operational problems that have arisen this week

The initial product is completed within the next week and tested and adjusted, continuously improving

Form Completed by: Yufeng Tao

Group Project Progress and Action Plan Sheet

Note: This form should be completed in electronic format but a hard copy should be printed out for submission at the end of each PIC Project lab session.

Date:2022/9/21

Group members present: Xukang Liu

Yufeng Tao

Ziyue Ren

Xiye Wen

Long Chen

1. Progress during this week

The plans made last week are divided in more detail and planned to the specific tasks of individuals. Complete the following steps: Step 1 into the system, the LCD displays "Enter Key", step 2 enter the set 4-digit password and store it, Step 3 sets 4 probe areas through KEYPAD, such as setting "1,2,3,4" for the probe area, step 4 to set the active area and the inactive area, such as setting "1,2" for the active area and "3,4" for the inactive area.

2. Problems encountered

The activation areas of step 4 and step 5 show that there is a problem, so you need to look for errors in the code to find the problem

Most of the code will run the basic requirements, but there are some minor issues that need to be identified and refined

There is no good clue and method for step 7 to solve the buzzer alarm problem, and it is necessary to further query the information to find a solution

3. Objectives for next week

Complete the following steps: Step 5 displays the active and inactive areas via LCD, Step 6 presses the button in the active area again, the security alarm device triggers, the buzzer alarm, and the step 7 LDC shows the trigger area above.

Complete debugging, modification, and improvement of some functions . Find out how to effectively lift the buzzer alarm and try.

Form Completed by: Yufeng Tao

Group Project Progress and Action Plan Sheet

Note: This form should be completed in electronic format, but a hard copy should be printed out for submission at the end of each PIC Project lab session.

Date:2022/9/28

Group members present: Xukang Liu

Yufeng Tao

Ziyue Ren

Xiye Wen

Long Chen

1. Progress during this week

This week, the entire group came to the school to do their own work, and each member gave their own opinions on how the PIC would work. During debugging this week, a program vulnerability was found in steps 3 and 4 completed last week that prevented the program from running properly, so this week detailed modifications and improvements were made to steps 3 and 4 to enable the program to run independently. At the same time, part of the code of the buzzer alarm has been completed, and the buzzer alarm has been able to run independently.

2. Problems encountered

The buzzer can run independently when only the buzzer alarm program is run, but when the program is placed in the entire program, the buzzer does not function properly

There is a partial bug in the program run, but the bug is not found through a single code run

Some members have problems with the task of perfecting the code and do not know how to do it

3. Objectives for next week

Complete step 6 so that the code for the buzzer alarm works correctly in the main program

Complete Step 7: The trigger area is shown on the LCD and Step 8: Use the password to remove the alarm from the buzzer

Form Completed by: Yufeng Tao

Group Project Progress and Action Plan Sheet

Note: This form should be completed in electronic format, but a hard copy should be printed out for submission at the end of each PIC Project lab session.

Date:2022/10/12

Group members present: Xukang Liu

Yufeng Tao

Ziyue Ren

Xiye Wen

Long Chen

1. Progress during this week

This week, our team has completed all its tasks and the PIC is up and running and rehearsing. The entire process of the PIC operation run was recorded because of the exercise. At the same time, the general framework of the PPT during the demonstration is prepared, which is divided into an introduction to the PIC function, an explanation of the code part and a final summary. Each member has their own demo section and division of tasks.

2. Problems encountered

The final report required a lot of space, which was quite different from the initial expectations of this group, and required a re-planning of the task The PIC runtime needs to have a function that can be returned to ensure that the user can return to the previous operation if it is wrong

Complete flowcharts are required in PPT and Final reports

3. Objectives for next week

Improve the PIC function and add a function that can be returned
Draw a flowchart of the PIC runtime

Improve the PPT, fill each plate completely

Complete the requirements of the final report and write the entire task process in the report

Form Completed by: Yufeng Tao