JIEE Časopis priemyselnej elektrotechniky
Journal of Industrial Electrical Engineering
ISSN 2454-0900
www.jiee.eu

Security alarm system

[1]*Simona KIREŠOVÁ,* [2]*Martina ZUBKOVÁ,* [3]*Jakub ŠIMČÁK,* [4]*Ján Molnár,* [5]*Patrik JACKO*

[1] Department of Theoretical and Industrial Electrical Engineering, FEI TU of Košice, Slovak Republic

[1]simona.kiresova@student.tuke.sk, [2]martina.zubkova@student.tuke.sk, [3]jakub.simcak@student.tuke.sk, [4]jan.molnar@tuke.sk, [5]patrik.jascko.2@tuke.sk

***Abstract* — The article is focused to solving the task of designing, programming and building a home security system. The system is controlled using by an Arduino board (to be specific, Arduino MEGA 2560) to implement the program, with a magnetic sensor to detect whenever someone enters the room. The code will be programmed using the Arduino IDE Software that is available on the official Arduino website.**

***Keywords* — Arduino MEGA 2560, LCD display, magnetic sensor, membrane keypad, piezo buzzer, security alarm system**

I. INTRODUCTION AND A BRIEF DESCRIPTION

The most basic definition of any security system can be found in its name. It is a means or a method of securing the property or belongings through a system of interworking components and devices working together to protect against burglars or other intruders. Security systems work on the simple concept of securing entry points (doors or windows) into a home with sensors that communicate with a control panel or command center (in this case, with *Arduino MEGA 2560* board) installed in a convenient location. Open spaces inside of the rooms can be secured by motion sensors, however, for this purpose they were not used and a magnetic sensor was chosen instead. Although if need be, it would be – with some level of alteration to the code – possible to use a motion sensor securing the open spaces (such as PIR sensor, for example).

Piezo buzzer was used as an alarm which goes off five seconds after the *magnetic sensor* detects that the door or the window was opened.

The *4x4 membrane keypad* serves the purpose of activating or deactivating the security system.

Moreover, the *16x2 LDC display* is used for displaying the instructions or information about the current state of the security system to makes interaction with the product more user-friendly.

The final part are three LED diodes - green, red, and yellow – that also convey information about the immediate state of the security system.

II. COMPONENTS AND SCHEMATICS

Arduino MEGA2560 is a development board based on the ATmega2560 microcontroller. It has 54 digital input/output pins (of which 5 can be used as PWM outputs) and 16 analogue inputs. It contains everything needed to support the microcontroller. It can be powered either by connecting to a computer with a USB cable or a battery (7 to 12V being the recommended input voltage). Arduino MEGA 2560 board can be programmed with the Arduino IDE Software. This platform was chosen for the security alarm system precisely because it has more digital input or output pins than Arduino UNO.

Arduino UNO would not have enough of the digital input/output pins to support this project, seeing as Arduino UNO only has 14 of them, while there are 19 digital input/output pins needed for this project, and therefore Arduino MEGA 2560 board was used instead.

16x2 LCD display with HD47780 driver has 16 pins. The first one from left to right is the ground pin. The second is the V_{CC} pin, connected to 5V output pin on the Arduino.

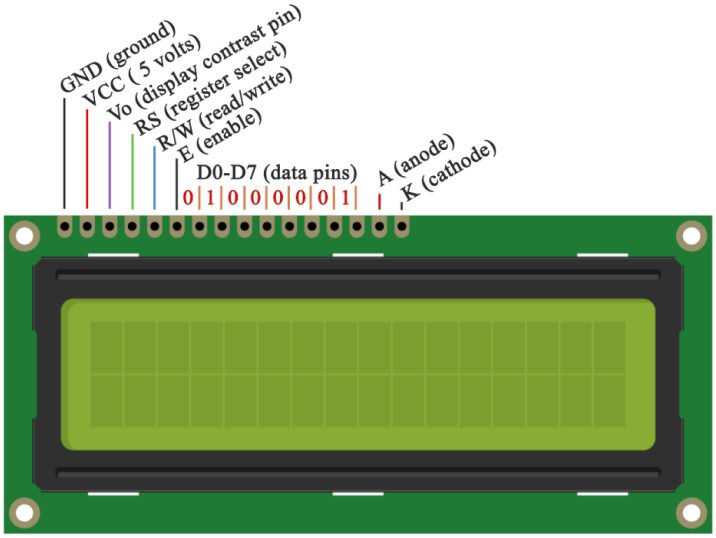

Fig. 1. A 16x2 LCD display and its pins

The next is the V_0 pin that controls the contrast of the display as long as a potentiometer or a voltage divider made of two resistors is connected to it. The latter option was chosen for this project. The values of the resistors are 220Ω and 1kΩ.

Next, the *RS* or register select pin is used for selecting whether to send commands or data to the LCD display. If the pin is set on low state, then the pin is sued for sending commands such as: set the cursor, clear the display, turn off the display, etc. When the pin is set on high state, the pin is used for sending data (characters, words) to the display.

The *R/W* pin selects whether the LCD display is in the read or write mode.

The *E* pin enables writing to the registers, or the next 8 data pins from D0 to D7. These 8 pins are used for sending 8 bits of data. Although for this project, only 4 of the 8 pins will be used, which means the LCD display will be used in a 4 bit mode.

The last two pins – *A* and *K*, or anode and cathode – are for the backlight of the display. To control and send data to the LCD display, the functions of the Liquid Crystal Library will be used.

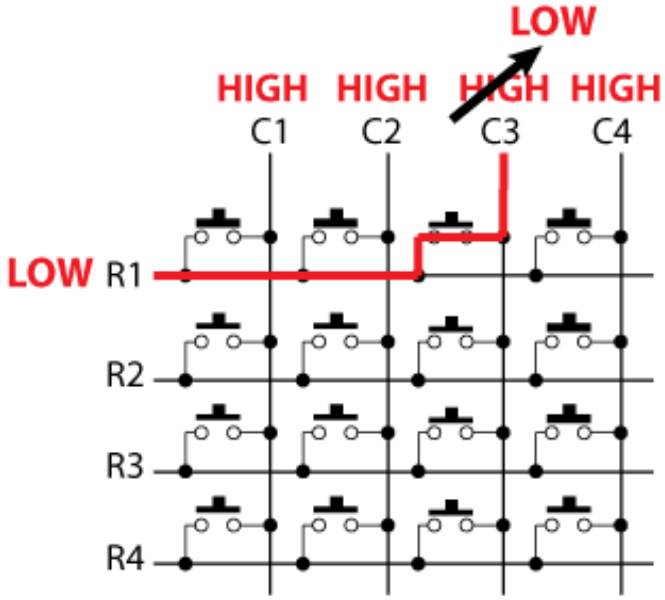

Fig. 2. Inner workings of a 4x4 membrane keypad

4x4 membrane keypad has eight pins – four are to rows and another four are for the columns. Each button is a push button switch. When no buttons are pressed, all the columns are set for high and all

the rows are set for low. When a button is pressed, the push button switch makes a short between a row and a column, which causes the voltage for the column to drop. This is how Arduino detects which column has been pressed.

Arduino detects which row has been pressed by setting each row to high again and at the same time reading all the column pins. When the column pin goes to high again, Arduino is able to find the row pin that is connected to the button. However, to be able to interface with the keypad, one has to additionally download the *Keypad.h* library which is not automatically included in the Arduino IDE Software.

Piezo buzzer is based on the inverse principle of piezo electricity – the phenomena of generating electricity when mechanical pressure is applied to certain materials and vice versa. When subjected to an electric field they stretch or compress in accordance with the frequency thereby producing sound. To create sound using Arduino, there are *tone()* and *noTone()* functions.

To detect whether the door or the windows have been opened, the *magnetic sensor* is used. Almost all door and window sensors use a *reed switch* to determine when a protected area has been breached. The two contacts are normally snapped together and when the magnet is brought up to the switch, they spring apart. Reed switches like this are called *normally closed.* Therefore, whenever the door opens – and the magnet moves away – the two contacts snapped together allow the current to flow and set the voltage from low to high. The magnetic sensor is also wired in series with 1kΩ resistor to prevent the damage.

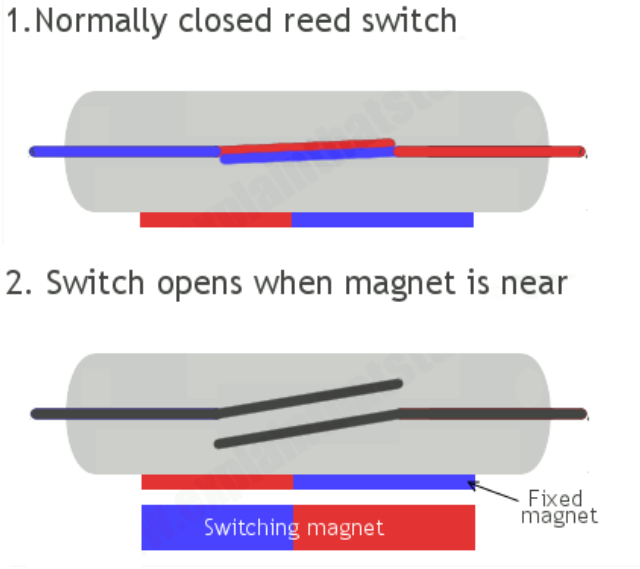

Fig. 3. A diagram demonstrating how the reed switch works

Last but not least, three *LED diodes* were used – yellow, green, and red. They convey information about the immediate state of the security alarm system. When the yellow LED is on, it means that the system has not yet been activated. Green LED indicates, that the system has been activated and the magnetic sensor is constantly checking whether the door has been opened. Red LED will turn on only if the intruder has entered the area and the alarm went off. All three LED diodes are connected to 220Ω resistors through their anode to prevent damaging them.

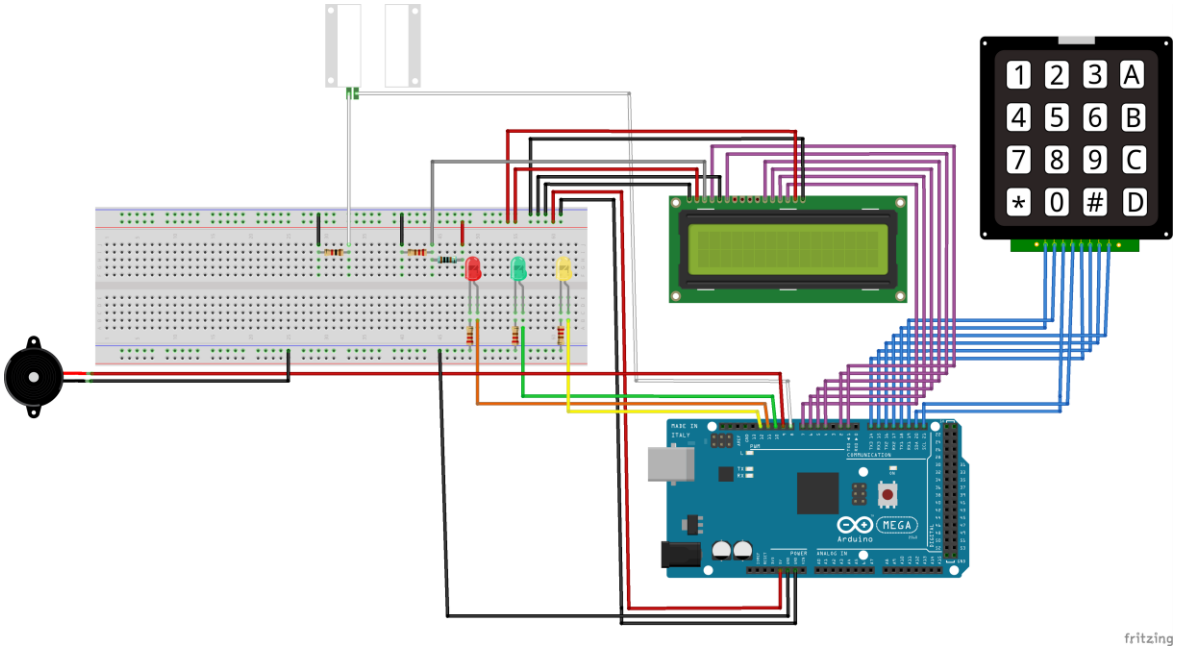

Fig. 4. Wiring of the components for the security alarm system designed with program Fritzing.

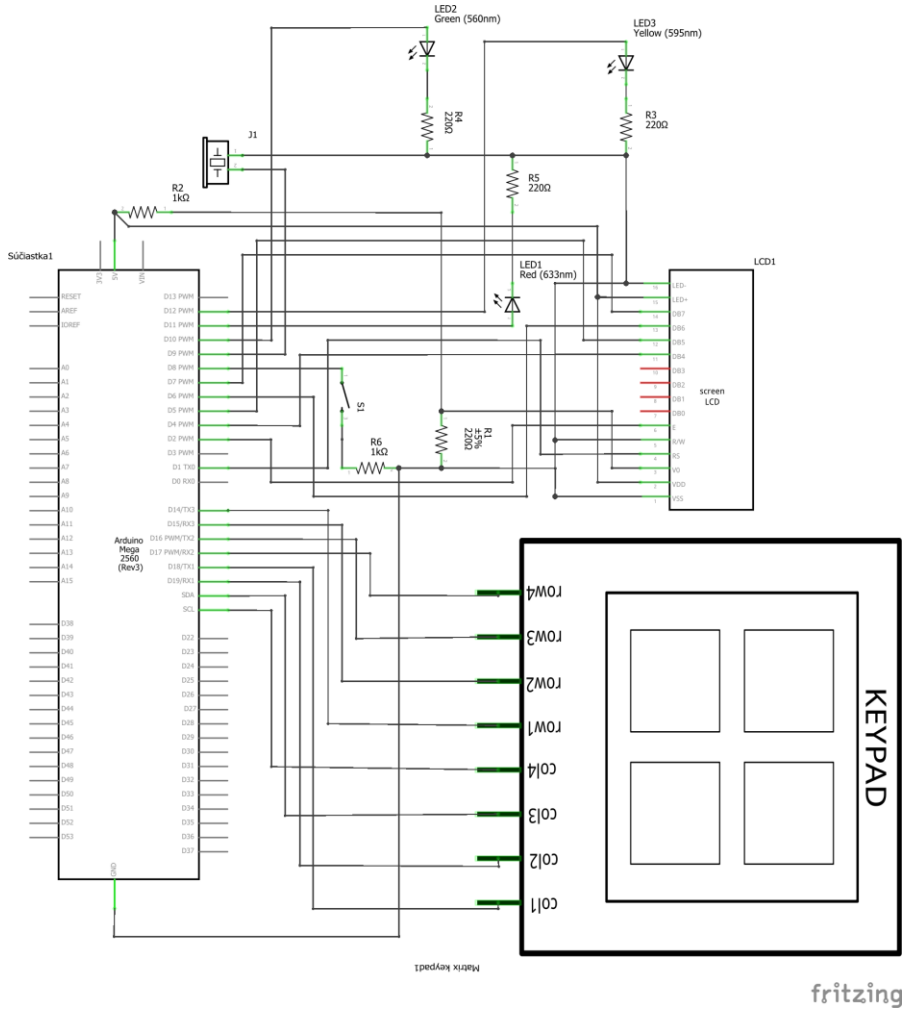

Fig. 5. An electrical schematic of the circuit designed with program Fritzing.

III. THE CODE AND HOW THE SYSTEM WORKS

```
#include <Key.h>
#include <Keypad.h>
#include <LiquidCrystal.h>
#define buzzer 9
#define sensor 8
#define redled 11
#define greenled 10
#define yellowled 12
int i,initialTime,currentTime;
String password=3D"2580";
String temporary;
boolean activated=3Dfalse;
boolean activateAlarm=3Dfalse;
boolean alarmActivated=3Dfalse;
boolean doorOpened=3Dfalse;
const byte ROWS =3D 4;
const byte COLS =3D 4;
char keypressed;
char keyMap[ROWS][COLS] =3D {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] =3D {14, 15, 16, 17};
byte colPins[COLS] =3D {18, 19, 20, 21};
Keypad myKeypad =3D Keypad(makeKeymap(keyMap), rowPins, colPins, ROWS, COLS);
LiquidCrystal lcd(1, 2, 4, 5, 6, 7);
```

First and foremost, the *Liquid Crystal* and *Keypad* libraries need to be included. While the Arduino IDE Software has a *Liquid Crystal* included, the *Keypad* library needs to be additionally downloaded and installed. Next, the pins of the magnetic sensor, buzzer and three LED diodes are defined, as to make the code more readable and easily editable. We also declare the variables we will be using later in the code, as well as create the keypad and LCD display objects.

```
void setup(){
  lcd.begin(16,2);
  pinMode(buzzer,OUTPUT);
  pinMode(sensor,INPUT_PULLUP);
  pinMode(redled,OUTPUT);
  pinMode(greenled,OUTPUT);
  pinMode(yellowled,OUTPUT);
  digitalWrite(redled,HIGH);
  digitalWrite(greenled,HIGH);
  digitalWrite(yellowled,HIGH);
  lcd.setCursor(1,0);
  lcd.print("SECURITY ALARM");
  lcd.setCursor(5,1);
  lcd.print("SYSTEM");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Made by:");
  delay(1000);
  lcd.setCursor(0,1);
  lcd.print("Kiresova Simona");
  delay(3000);
  lcd.setCursor(0,1);
  lcd.print("Simcak Jakub    ");
  delay(3000);
  lcd.setCursor(0,1);
  lcd.print("Zubkova Martina");
  delay(3000);
    }
```

The *setup()* function initializes pins of the three LED diodes as well as the pin of the buzzer as outputs. The pin of the magnetic sensor is initialized as an input.

Next, LED diodes are turned on. LCD display prints out the words ‘SECURITY ALARM SYSTEM’ and then it prints out the credit to the creators of this alarm system.

Next comes the loop function, explanation of which will be split into several parts.

```
void loop() {
if (activateAlarm) {
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Alarm will be");
  lcd.setCursor(0,1);
  lcd.print("activated in");
  int countdown=10;
  while(countdown!=0){
    lcd.setCursor(13,1);
    lcd.print(countdown);
    if(countdown!=10){
     lcd.setCursor(14,1);
     lcd.print(" ");
    }
    countdown--;
    tone(buzzer, 700, 100);
    delay(1000);
  }
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Alarm Activated!");
  activateAlarm=false;
  alarmActivated=true;
}
```

The program starts by checking if the variable *activateAlarm* has been set to true. If that is the case, then the countdown of ten seconds will start, giving us time to exit the room before the system is activated. During the countdown, the LCD display prints out a message that informs us how much time we have left. Furthermore, each passing second is marked by a 700Hz sound made by the piezo buzzer. After the countdown has reached zero, the alarm will activate by setting the *alarmActivated* variable to true.

```
if(alarmActivated==true){
    digitalWrite(greenled,HIGH);
    digitalWrite(redled,LOW);
    digitalWrite(yellowled,LOW);
    if(digitalRead(sensor)==HIGH){
      doorOpened=true;
    }
    if(!doorOpened) {
      initialTime = millis();
    }else if(doorOpened){
      currentTime=millis()-initialTime;
      if(currentTime<=10000){
        enterSoon();
      }else if (currentTime>10000 && alarmActivated==true){
        goesOff();
      }
    }
}
```

This part of the loop function checks whether the system has been activated. If that is the case, then firstly, the green LED diode will be on.

Next, the system will be checked if the door has been opened yet. If not, then the value of *doorOpened* variable will be *false*, as has been defined above the setup function. When the door opens and the magnetic sensor reads high, the value of that variable will change to *true*. However, even if the intruder closes the door and the sensor will read low again, the variable will not be changed back to false. The only way to do that will be to deactivate the system.

Had we not used this variable and only used the readings from the sensor directly, the intruder could have easily closed the door behind him and the alarm would not go off.

The reason for this is because the alarm does not go off immediately, but gives time of 10 seconds to input the password and deactivate it, in case it is not the intruder who enters the room, but the owner.

It works like that - if the door has not been opened yet, then the Arduino will be checking how much time has passed since the program started running with the *millis()* function and save that value into the *initialTime* variable. The final value of the variable will be the last one before the intruder has entered the room.

After the sensor pin reads high, and *doorOpened* changes from *false* to *true*, the program will constantly calculate how much time has passed since the room has been invaded, by subtracting the final value of the *initialTime* variable from the current running time and saving it into the *currentTime* variable.

This allows for the 10 second time window to enter the password. If, however, the password has not been entered within that timeframe the alarm will go off, which can be shut down only by entering the correct password.

```
if(!alarmActivated){
      digitalWrite(yellowled,HIGH);
      digitalWrite(greenled,LOW);
      digitalWrite(redled,LOW);
      activateWithPassword();
  }
}
```

The final part of the loop section will be covering the situation of the alarm not been activated yet. The yellow LED diode will be on indicating that state of the system. The *activateWithPassword()* function is called.

There are, however, four more void functions that are called in the loop function of the program. And although three of them are only used in the code once and the remaining one is used twice, we have chosen to make functions of these parts of code to make the loop function more readable and easily understandable.

```
void enterPassword(){
  i=9;
  temporary ="";
  activated = true;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Enter password:");
  lcd.setCursor(0,1);
  lcd.print("Password>");
      while(activated) {
      keypressed=myKeypad.getKey();
      if (keypressed!=NO_KEY){
        if (keypressed == '0' ||
            keypressed == '1' ||
            keypressed == '2' ||
            keypressed == '3' ||
            keypressed == '4' ||
            keypressed == '5' ||
            keypressed == '6' ||
            keypressed == '7' ||
            keypressed == '8' ||
            keypressed == '9' ){
               temporary += keypressed;
               lcd.setCursor(i,1);
               lcd.print("*");
               i++;
        }
      }
      if (i > 13 || keypressed == 'D'){
        temporary="";
        i=9;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Enter password:");
        lcd.setCursor(0,1);
        lcd.print("Password>");
      }
      if(keypressed=='#'){
        if(temporary==password){
          activated=false;
          alarmActivated=false;
          noTone(buzzer);
          doorOpened=false;
        }
        else if(temporary!=password){
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Wrong! Try Again");
          delay(2000);
```

```
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Enter password:");
          lcd.setCursor(0,1);
          lcd.print("Password>");
        }
      }
    }
  }
```

The first function is the *enterPassword()* function. It is used for deactivating the alarm, by entering the password (that has, in the beginning of the program above the *setup()* function, been defined as 2580).

While using the loop, the program is constantly checking whether a button on the keypad has been pressed, and each pressed button id added to the *temporary* variable. If more than 4 digits are entered, or if the 'D' (for 'delete') button is pressed, then the *temporary* variable will be cleared, and the password can be entered again.

To input the password, the sharp button needs to be pressed. Then the program will check whether the correct password has been entered. If that is the case, then the piezo buzzer will stop producing the sound and the system will deactivate.

However, if the password that has been input is incorrect, then the LCD display will inform us about it, the buzzer will continue producing the sound, and we are free to try entering the password again.

```
void enterSoon(){
     keypressed=myKeypad.getKey();
     if(keypressed=='2'){
       temporary=keypressed;
       lcd.clear();
       lcd.setCursor(0,0);
       lcd.print("Enter password:");
       lcd.setCursor(0,1);
       lcd.print("Password>*");
       i=10;
       activated = true;
       while(activated&&currentTime<=10000){
         keypressed=myKeypad.getKey();
       currentTime=millis()-initialTime;
  if(currentTime>10000&&alarmActivated==true){
                goesOff(); }
           if (keypressed!=NO_KEY){
             if (keypressed == '0' ||
                 keypressed == '1' ||
                 keypressed == '2' ||
                 keypressed == '3' ||
                 keypressed == '4' ||
                 keypressed == '5' ||
                 keypressed == '6' ||
                 keypressed == '7' ||
                 keypressed == '8' ||
                 keypressed == '9' ){
                   temporary+=keypressed;
                   lcd.setCursor(i,1);
                   lcd.print("*");
                   i++;
             }
           }
           if(i > 13 || keypressed=='D'){
             temporary="";
             i=9;
             lcd.clear();
             lcd.setCursor(0,0);
             lcd.print("Enter password:");
             lcd.setCursor(0,1);
             lcd.print("Password>");
           }
           if(keypressed=='#'){
             if(temporary==password){
               activated=false;
               alarmActivated=false;
               doorOpened=false;
            }else if(temporary!=password){
              lcd.clear();
              lcd.setCursor(0,0);
              lcd.print("Wrong! Try Again");
```

```
              tone(buzzer, 600, 100);
              delay(500);
              tone(buzzer, 600, 100);
              delay(1500);
              lcd.clear();
              lcd.setCursor(0,0);
              lcd.print("Enter password:");
              lcd.setCursor(0,1);
              lcd.print("Password>");
            }
          }
        }
      }
}
```

This function is used for deactivating the alarm system within the timeframe of 10 seconds before the alarm goes off. Since the circumstances are different here than in the situation where the alarm has already been activated, the *enterPassword()* function could not be used in its original form, and therefore we had to make another, altered version of it, the result of which is the *enterSoon()* function.

While we do need to input the same password and check whether it matches with the password that is set for this system, there are also a few differences that had to be made.

The first and most obvious one is that the function must check whether or not have 10 seconds passed or not and in case that yes, it has already been 10 seconds, the alarm needs to be activated. We have also used a similar part of code in the loop function.

The second difference is more cosmetic in nature, that being that the message about entering the password to deactivate the system will appear only after the first correct digit of the password has been pressed.

Entering and comparing the password works similarly to *enterPassword()* function.

Another addition to that is that if the password that has been entered is incorrect, then in addition to LCD display printing out that information and the program letting us try entering the correct password again, the buzzer will also produce two consecutive 600Hz sounds that last 100 milliseconds.

It even gives you the chance to enter the password again, if the first try has been incorrect, however, due to the 10 seconds time limit, it is improbable you will be able to enter the password again before the time runs out.

It is important to note that if the *enterPassword()* function has been used instead of *enterSoon()*, the alarm would never go off which would render the whole system quite useless.

```
void activateWithPassword(){
  i=12;
  temporary="";
  activated=false;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Enter password");
  lcd.setCursor(0,1);
  lcd.print("to activate>");
      while(activated==false) {
      keypressed=myKeypad.getKey();
      if(keypressed!=NO_KEY){
        if (keypressed == '0' ||
            keypressed == '1' ||
            keypressed == '2' ||
            keypressed == '3' ||
            keypressed == '4' ||
            keypressed == '5' ||
            keypressed == '6' ||
            keypressed == '7' ||
            keypressed == '8' ||
            keypressed == '9' ) {
              temporary+=keypressed;
              lcd.setCursor(i,1);
              lcd.print("*");
              i++;
        }
      }
      if(i > 16 || keypressed == 'D'){
        temporary="";
        i=12;
        lcd.clear();
        lcd.setCursor(0,0);
```

```
        lcd.print("Enter password");
        lcd.setCursor(0,1);
        lcd.print("to activate>");
      }
      if(keypressed=='#'){
        if(temporary==password){
          activated=true;
          activateAlarm=true;
        }
        else if(temporary!=password){
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Wrong! Try Again");
          tone(buzzer, 600, 100);
          delay(500);
          tone(buzzer, 600, 100);
          delay(1500);
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Enter password");
          lcd.setCursor(0,1);
          lcd.print("to activate>");
        }
       }
     }
}
```

The third void function is the *activateWithPassword()* function. It is used for activating the system with a password, which means, that it, again, shares its core with the *enterPassword()* function, even more so than the *enterSoon()* function does.

Entering and comparing the *temporary* with *password* stays the same. The difference, however, comes in, after the sharp button is pressed to confirm whether the correct password has been entered. If that is the case, instead of deactivating the alarm, we activate the alarm instead.

Similarly to the *enterSoon()* function, if the password that has been entered is incorrect, then the LCD display will print out the message informing you of this as well as the buzzer will produce two consecutive 600Hz sounds that last 100 milliseconds. The *enterPassword()* function lacks this because since it is used only after the alarm has already gone off, the piezo buzzer is already in use.

```
void goesOff(){
  digitalWrite(redled,HIGH);
  digitalWrite(greenled,LOW);
  digitalWrite(yellowled,LOW);
  tone(buzzer,1000);
  lcd.clear();
  enterPassword();
}
```

The final void function is called *goesOff()*. As its name suggests, it causes the alarm to go off by activating the piezo buzzer at a 1kHz frequency. The red LED diode turns on to convey the information about the current state of the alarm system. It then calls another void function, namely *enterPassword()*, which is the reason why it's so short.

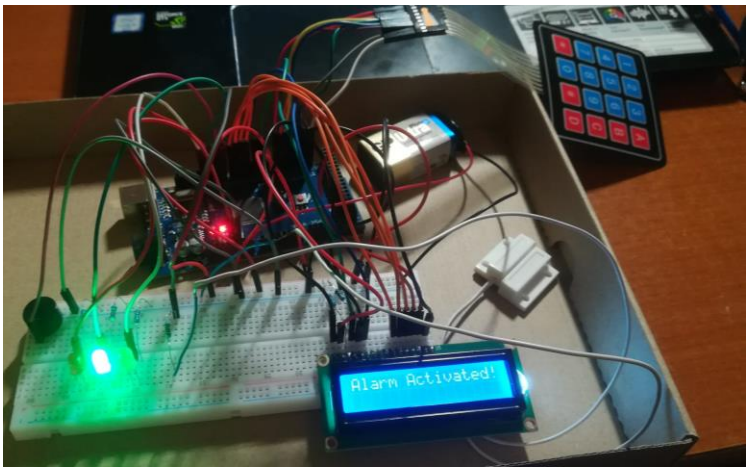

Fig. 6. The security alarm system in the developing stage.

Fig. 7. The finished project.

IV. CONCLUSION

The Arduino board is useful programmable device providing comfortable programing and developing prototypes. This topic talked about Arduino board and its using to the security alarm system. However, in the extended of this device is possible include a lot of another components and makes more secure system by using ultrasonic sensor, PIR sensor etc.

ACKNOWLEDGMENT

We support research activities in Slovakia / Project is co-financed from EU funds. This paper was developed within the Project "Centre of Excellence of the Integrated Research & Exploitation the Advanced Materials and Technologies in the Automotive Electronics ", ITMS 26220120055

REFERENCES

[1] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.
[2] W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.
[3] H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.
[4] B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.
[5] E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.
[6] J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.
[7] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.
[8] Arduino DUE [online]. [cit. 2017-4-2] < https://www.arduino.cc/en/Main/arduinoBoardDue>
[9] Guzan M., Špaldonová D., Hodulíková A., Tomčíková I., Gladyr A.: Boundary Surface and Load Plane of the Ternary Memory, In: Electromechanical and energy saving systems. Vol. 15, no. 3 (2011), p. 163-167. - ISSN 2072–2052
[10] Bereš M., Perduľak J., Kováč D.: Autonomous mobile robot with obstacles prediction In: SSIEE 2014 : proceeding of scientific and student's works in the field of Industrial Electrical Engineering : volume 3. - Košice : TU, 2014 S. 166-169. - ISBN 978-80-553-1711-4

[11] Kováčová I., Kováč D., Vince T.: Elektromagnetická kompatibilita - 1. vyd - Košice : TU, - 2009. - 137 s. - ISBN 978-80-553-0150-1.
[12] Bučko,R., Kováč,D., Konokh,I.: Embedded systém and speech recognition, Electromechanical and energy saving systems: Quarterly research and production journal Vol. 2011, No. 3 (2011), pp. 168-172, ISSN 2072-2052.
[13] Dziak,J. : Linear circuit simulation using MATLAB and modeling of nonlinear elements, In: SCYR 2014 Proceeding from Comference: 20.5.2014: Herľany, S. 70 - 71, Košice : Technická univerzita v Košiciach, 2014 /978-80-553-1714-4/.
[14] Jacko P., Kováč D.,: Converters and time conversion measurement of STM32F446RE microcontroller , In: Electromechanical and energy systems. Modeling and optimatization methods. - Kremenchuk : Kremenchuk Mykhailo Ostrohradskyi National University, 2017 P. 154-155. - ISSN 2079-5106.