Electronic Technologies Engineering

Telecommunications Project 243-699 Winter 2023

Security level-based lock

User Manual

Prepared By: Ibrahim Elyadini

Date: May 10th, 2023

**Purpose of the security level-based lock:**

The main idea behind this project was the idea of having “information” that depending on its sensitivity would be locked behind a certain level of security. This concept was then later adapted in the following manner where security access was locked behind the unlocking method depending on how hard it would be for a bad actor to obtain it. So, then the hierarchy was made where RFID tags were attributed the lowest level of access, the keypad was attributed the second level access, and finally the frequency tone detector was given the highest level of access. Based on the access level, users could unlock one or more solenoids.

**Brief technical description:**

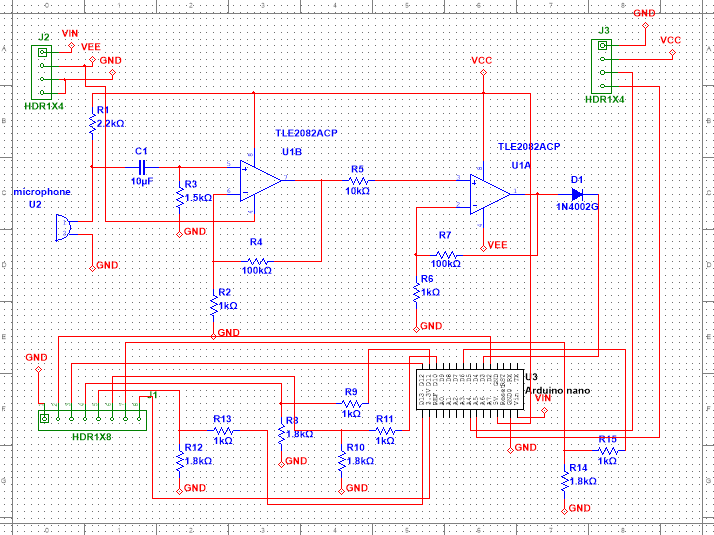
The Arduino Uno is connected to the RC522 and will turn on one of the solenoids whenever an authorized RFID tag that is recorded makes contact with the reader. The Arduino Uno can also grant access and remove access by firstly reading the master key card and then reading a tag. The 8515 is connected to the keypad, it reads the keypad continuously until a valid key press is entered and then adds it to an array where it keeps track of the user’s attempt and after 4 key presses it compares it with another array which contains the right code and will turn on the previous solenoid as well as another one whenever it receives the appropriate sequence of signals from the keypad. The other board which contains the microphone circuit will amplify the original signal until it reaches the limits (5V to -5V) and then a diode will be placed between the signal and the Arduino nano to be able to analyze the signal. If the correct tone is played it will send out wirelessly data to the other Arduino nano and turn on all the solenoids. Near the solenoids there is also a speaker which will output a tone whenever a solenoid is turned on. An OR gate is responsible for choosing which solenoids are unlocked when access is granted as well as turn on the 555 timer that produces the tone for the speaker.

Below you will find explanations that go more in depth on each module with schematics, flowcharts, and code sample.

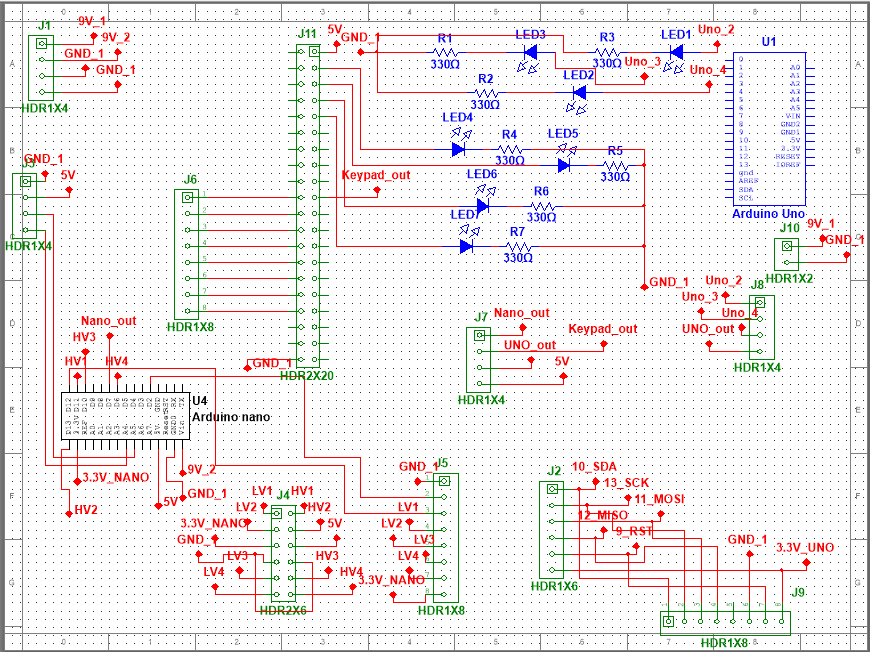
**Guide of operation:**

When the circuits have been properly set up the user should be able to make use of the functionality of the product. Firstly, they should try the RFID module after having everything powered on. If this is the first time using this product the user should initially scan the PICC card to get a master key card. Afterwards they will be able to add and remove users at will. Try adding a user by first scanning the master key card then scanning an unknown RFID tag. You will know it is successful if the LED starts blinking after you scanned it. Then leave this mode by rescanning the master key card and scan your newly added key card and you should hear the speaker produce a tone as well as see the first solenoid unlock. Moving onto the Keypad module, if you wanted to change the password all you would have to do is go into the code snippet’s part and change the array which contains the code which is currently setup as 1,2,3,4 to whatever code you desire. Now after changing the code, you will be able to test your code by inputting using the keypad. Be mindful of the time between each keypress, there is a 1 second delay between each key presses so wait for the red LED to light up again before pressing another button. When the orange LED is on that means you need to wait. And when the blue LED is on that simply means you sent out an invalid press which can be the usage of the 4th column keys or pressing 2 keys at the same time. If you got the right code, you should see a green LED light up as well as hear the speaker and see the first and second solenoids unlocked. Now moving onto the third and last module which is the tone frequency detection, you’re going to want to change the tone to a frequency you desire by changing the if statement that compares the frequency to whatever value you want. In order to operate this module, you’re going to want to play the tone relatively close to the mic with a volume that is also relatively big since the microphone is not very sensitive. The LCD display will show you the tone frequency being played on it, and if the correct tone is played a signal will be sent via the CC1101 and the LCD will notify you. On the other hand of the transmission, if the transmitted signal is appropriate, the speaker will produce a tone and all the solenoids will unlock. Be mindful of the distance between the CC1101 as the transmission range is relatively limited.

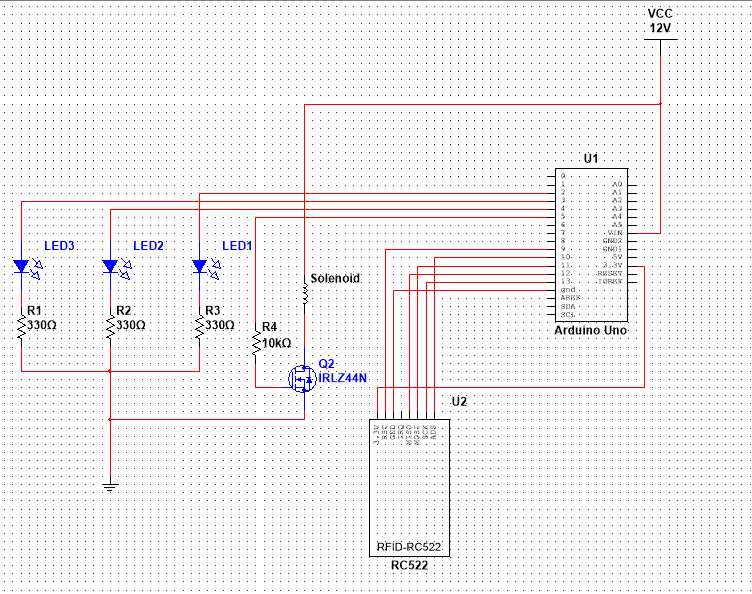
**Schematics:**



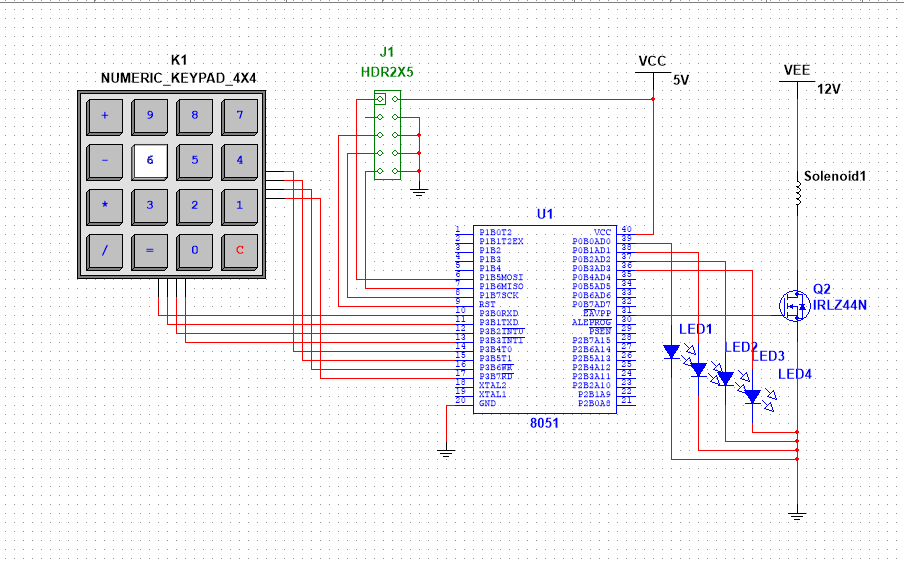
*Figure 1.1: Board 1 schematic*



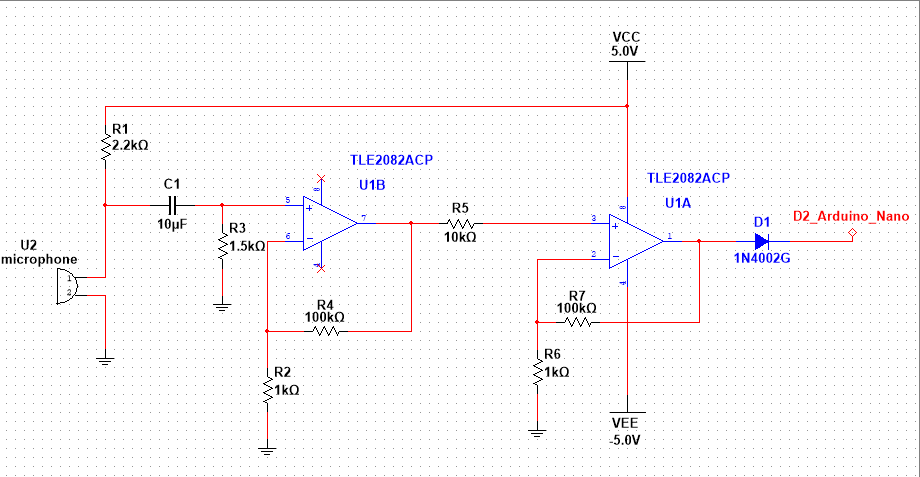
*Figure 1.2: Board 2 schematic*



*Figure 1.3: RFID module*



*Figure 1.4: Keypad module*



*Figure 1.5: tone detection module*

*A diagram of a circuit board

Description automatically generated with low confidence*

*Figure 1.6: speaker circuit*

*A picture containing screenshot, line, colorfulness

Description automatically generated*

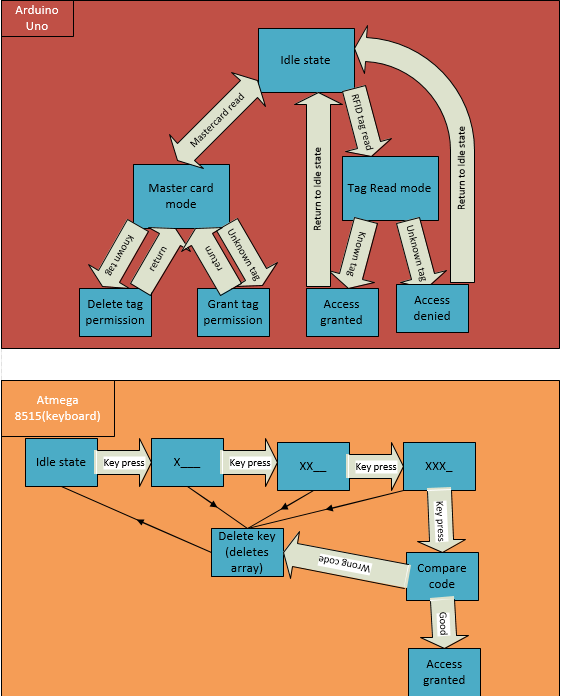
*Figure 1.7: Board 1 PCB layout*

*A computer screen shot of a circuit board

Description automatically generated with low confidence*

*Figure 1.8: Board 2 PCB layout*

**Flowchart:**



*Figure 2.1: RFID module flowchart*

A picture containing text, screenshot, diagram, parallel

Description automatically generated

*Figure 2.2: keypad module flowchart*

A picture containing text, diagram, screenshot, plan

Description automatically generated

*Figure 2.3: tone detection module flowchart*

**Code snippets:**

The code for the RFID module was taken from the access control exemple from the MFRC522 library on Arduino and was modified. Here are the relevant changes:

A screenshot of a computer program

Description automatically generated with medium confidence

Lock delay is set to 5 seconds and the pins for the LED and wipe and relay were switched to be able to interface the RFID reader.

A picture containing text, font, screenshot

Description automatically generated

The value of the relay were swapped so instead of outputting a high when it is locked, it outputs a low and that allows us to save a lot more on power consumption.

A screenshot of a computer program

Description automatically generated with low confidence

Same here we can see the relay become high when access is granted.

//////////////////////////////////////////////////////////////////////////////////////////////////////////////

The code for the keypad was done on the atmega8515 using C here is the code.

/\*

\* Keypad\_Operation.c

\*

\* Created: 1/30/2023 11:39:20 AM

\* Author : 2038471

\*/

#include <avr/io.h>

#define *F\_CPU* 1000000UL

#include <util/delay.h>

#define readc1 0x01 ; //read column 1 input

#define readc2 0x02 ; //read column 2 input

#define readc3 0x04 ; //read column 3 input

#define readc4 0x08 ; //read column 4 input

void getkey();

void checkkey();

void checkcode();

void outputkey();

void outputkeyc1();

void outputkeyc2();

void outputkeyc3();

void deletecode();

int code[5]={1,2,3,4}; //the numbers on the keypad activate other keypad numbers after they have been pressed ex: 6=4, 9=7, 3=?

int samplecode[5]; //maybe use char

int previouskeypress;

int keypressvar;

int insc[4]; //maybe just 3

int x;

int y; //used for the array of the sample code

int led;

int main(void)

{

//DDRD = 0x00; //Keypad input

DDRC = 0xFF; //output to see number

DDRA = 0xFF;

DDRE = 0xFF;

PORTE=0x00;

x=0;

y=0; //initialized

previouskeypress = 0;

keypressvar = 0;

while (1)

{

getkey();

checkkey();

}

}

void getkey()

{

DDRD = readc1; //first column

PORTD = 0xF0; //set as low to read

insc[0] = PIND;

insc[0]=insc[0]>>4;

PORTD = 0x00; //reset output

DDRD = 0x00; //reset input

DDRD = readc2; //Second column

PORTD = 0xF0; //set as low to read

insc[1] = PIND;

insc[1]=insc[1]>>4;

PORTD = 0x00; //reset output

DDRD = 0x00; //reset input

DDRD = readc3; //third column

PORTD = 0xF0; //set as low to read

insc[2] = PIND;

insc[2]=insc[2]>>4; //For testing purposes they are kept here

PORTD = 0x00; //reset output

DDRD = 0x00; //reset input

DDRD = readc4; //first column

PORTD = 0xF0; //set as low to read

insc[3] = PIND;

insc[3]=insc[3]>>4;

PORTD = 0x00; //reset output

DDRD = 0x00; //reset input

return;

}

void checkkey()

{

for(x=0;x<3;x++)

{

if(insc[x]==0x0F)

{

PORTA = 0x01;

}

else

{

outputkey();

return;

}

}

return;

}

void outputkey()

{

PORTA = 0x02;

switch(x)

{

case 0:outputkeyc1(); *\_delay\_ms*(1000.0); break;

case 1:outputkeyc2(); *\_delay\_ms*(1000.0); break;

case 2:outputkeyc3(); *\_delay\_ms*(1000.0); break;

}

if(y>3)

{

checkcode();

}

return;

}

void outputkeyc1()

{

switch(insc[0])

{

case 14: samplecode[y]=1; y++; break;

case 13: samplecode[y]=4; y++; break;

case 11: samplecode[y]=7; y++; break;

case 7: PORTA = 0x04; break; //not number

default: PORTA = 0x04; break;

}

return;

}

void outputkeyc2()

{

switch(insc[1])

{

case 14: samplecode[y]=2; y++; break;

case 13: samplecode[y]=5; y++; break;

case 11: samplecode[y]=8; y++; break;

case 7: samplecode[y]=0; y++; break;

default: PORTA = 0x04; break;

}

return;

}

void outputkeyc3()

{

switch(insc[2])

{

case 14: samplecode[y]=3; y++; break;

case 13: samplecode[y]=6; y++; break;

case 11: samplecode[y]=9; y++; break;

case 7: PORTA = 0x04; break;

default: PORTA = 0x04; break;

}

return;

}

void deletecode()

{

samplecode[0]=0;

samplecode[1]=0;

samplecode[2]=0;

samplecode[3]=0;

y=0;

x=0;

insc[0]=0x0F;

insc[1]=0x0F;

insc[2]=0x0F;

return;

}

void checkcode()

{

int error=0;

if(samplecode[0]!=code[0])

error++;

if(samplecode[1]!=code[1])

error++;

if(samplecode[2]!=code[2])

error++;

if(samplecode[3]!=code[3])

error++;

if(error==0)

{

PORTA = 0x08;

PORTE = 0xFF;

deletecode();

*\_delay\_ms*(5000.0);

PORTA = 0x00;

PORTE =0x00;

y=0;

return;

}

else

{

deletecode();

PORTA = 0x08;

*\_delay\_ms*(200.0);

PORTA = 0x04;

*\_delay\_ms*(200.0);

PORTA = 0x02;

*\_delay\_ms*(200.0);

PORTA = 0x01;

*\_delay\_ms*(200.0);

y=0;

return;

}

}

/////////////////////////////////////////////////////////////////////////////////////////

The code for the wireless transmission was done on the Arduino nanos and is relatively similar here are some code snippets.

A picture containing text, screenshot, font

Description automatically generated

Code for the frequency tone detection only relevant for the nano on board 1.

A picture containing text, screenshot, font, document

Description automatically generated

Code for set up of the modulation and transmission of data wirelessly through the CC1101 is common for both.

Bill of materials:

|  |  |
| --- | --- |
| Parts | amount |
| Atmega 8515 | 1 |
| keypad | 1 |
| speaker | 1 |
| microphone | 1 |
| I2C LCD | 2 |
| Arduino Uno | 1 |
| Arduino Nano | 2 |
| RFID reader | 1 |
| Resistors | \* |
| Power adapter (3.3V-12V) | 3 |
| 9V batteries | 3 |
| LEDs | 7 |
| capacitors | \* |
| usb cables (for programming) | 2 |
| Solenoids | 3 |
| CC1110 wireless transceiver | 2 |
| 555 timer | 1 |
| Op amp | 1 |
| diodes | 1 |
| N-channel mosfet | 3 |

\*Actual amount may vary and should be decided based on what the multisim schematic show.