Home security system

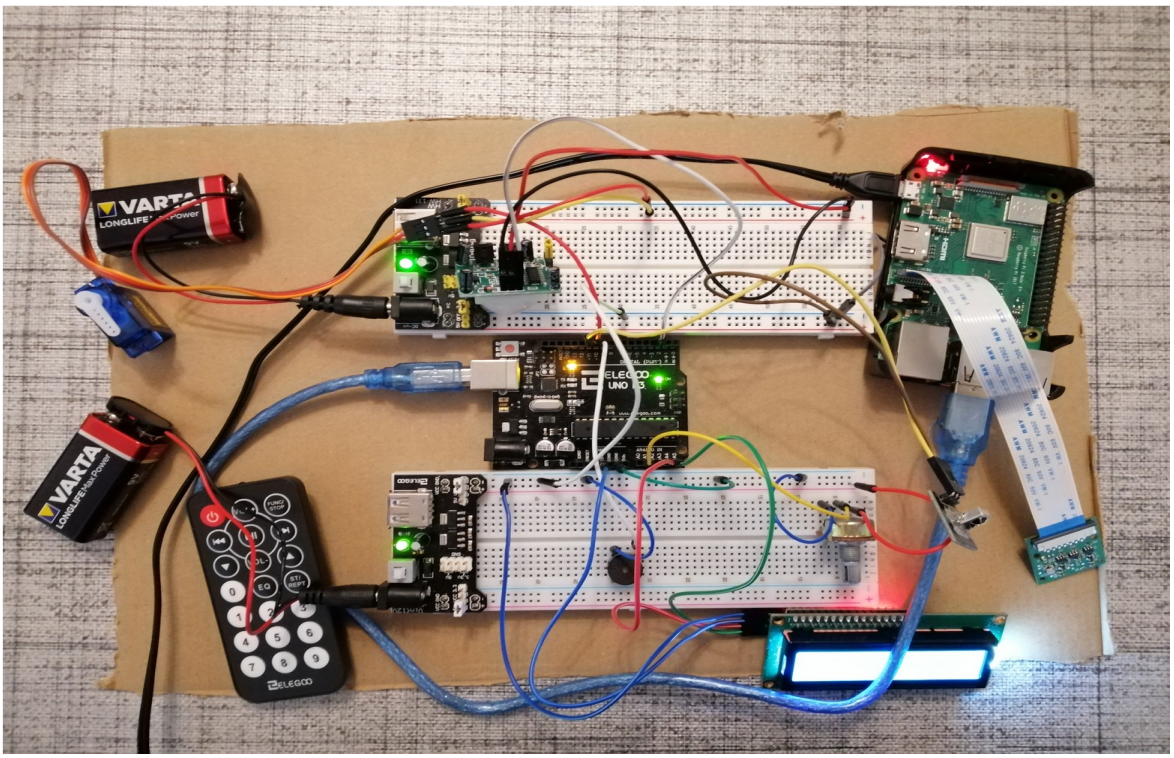
INTERNATIONAL UNIVERSITY OF SARAJEVO

Course : EE325

Student : Ejub Bilajbegovic

Professor : Tarik Namas

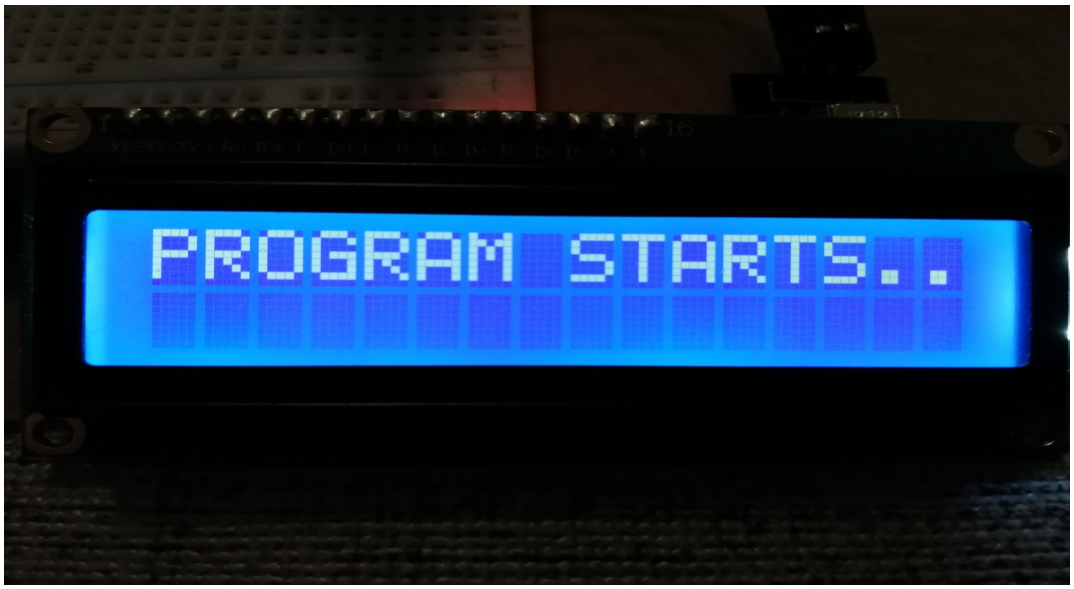
Date : 7 / 6 / 2020



Introduction

Overview

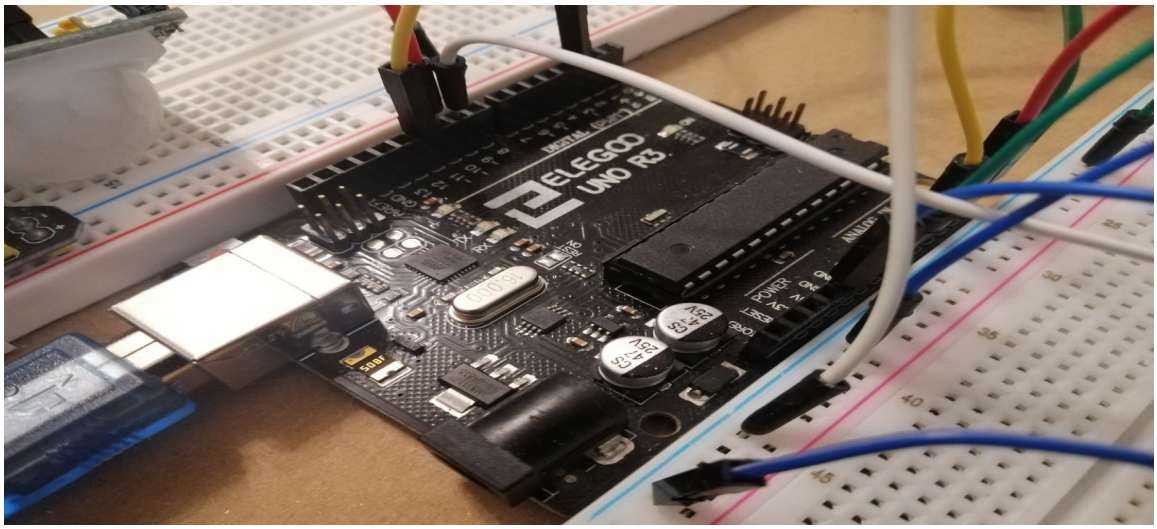
The main goal of a Home security system is to protect the house residents and give them safety. Although 100% safety is impossible , cause every human systems has errors and vulnerability , i tried my best with my knowledge and resources that is possesed . My home system consists of varius safety mechanism which when triggered will turn on the alarm and in some cases make photos to record evidence. My project also has a lcd user interface which is easy to use . It allows them to open the door by inserting their pins(the pin should be 10 numbers long) , if a wrong person tried to enter the house and entered it three times wrong the alarm will go on. A nice feature that i added is that i saved information on who opened and when the door , so more transperacy will be reached. When all residents are on vacation or in their bedrooms , they can turn the movement sensor on , which will when detected motion in the entrance room turn on the alarm. Of course in all situation the alarm could be turned off. An another protection is a smoke detector , which works all the time. Because not all people know programming and reading or writing from the terminal shell looks ugly and unprofessionell i designed also a simple graphical user intercafe . Its like a app where the user has to log in and then can make changes as turn off movement sensor.



Working Concept

The Alarm system consists of one single board computer called Raspberry pi model 3b+ and the arduion mircocontroller. I use the arduino for reading sensor value and controlling actuators in my project. Arduino then sends it to raspberry pi , there is a softwere written in arduino for the lcd interface and alarm system . So the arduino will turn on the alarm by itself , it doesnt need the raspberry pi for it , but it needs some instruction from the raspberry. The final part is the GUI what completes the project , after naming all my components I will start describing the working routine of my project. First we need to understand how the arduino works. In the arduino we read values from three sensors , for smoke , ir signals and movement detection. The smoke sensing is pretty easy , whenever it reads a signal higher than a procentage that we want , turn on the alarm. The movement detection works only when wanted from user , so the raspberry needs to be in that mode to read that sensor , for that we will read the instruction from the raspberry pi , if we get the instruction to turn the sensor on we will read it , otherwise we wont. To understand the ir reciver I will start describing the LCD interface since lcd , ir reciver and servo are connected and are a systen on its own. The ir reciver can get a number(0-9) , check button or quit button signal. The quit button turns on/ off the lcd , when turned off it will delete the current pin and restart the pin. If the lcd is on we can add numbers , the pin should be 10 digit long . If we inserted our pin good we can press the check button . Next the arduino will send the current pin to raspberry to check is the pin valid. If so it will open the door(The servo motor will move to unlocked) and lcd will output that password is right , else it will output its wrong. If someone inserted three wrong pins in a row then the alarm will go on. If the alarm is on the arduino will send a message to the raspberry. The connection between them is done using the USB ports. Thats nearly most of the arduino and I will continue to the raspberry. The raspberry is the upgrade to the code on arduino , yes it works on the arduino but its primitive and we need a data base , server , tacking pics … And thats why I use the raspberry pi. First we set the connection between the arduino and raspberry , if it works we chkeck the server and databeses . If everything works fine we can start with the routine. The database consists of two tables , one for the user data and one for the Entrance login information. The raspberry pi reads the messages from arduino , if alarm is tuned on it will recive the reason , and store it . If user opens GUI he will see that something is wrong. If we recive a message for the movement sensor the raspberry will make camera pics for a minute, and store it in a file on Desktop. Also if it recives the pin to check it will look in the Users table and if its true add new login in the Entrance login table. It will also have a sever which will always listen to clients. Raspberry pi will send information to the GUI or accept them. Lastly the GUI is there for easy use. First we need to identify that we are a user , for that I used a simple login dialog if its true a setting dialog will open. From there we can see the status and turn on/off movement option , which I called alarm.

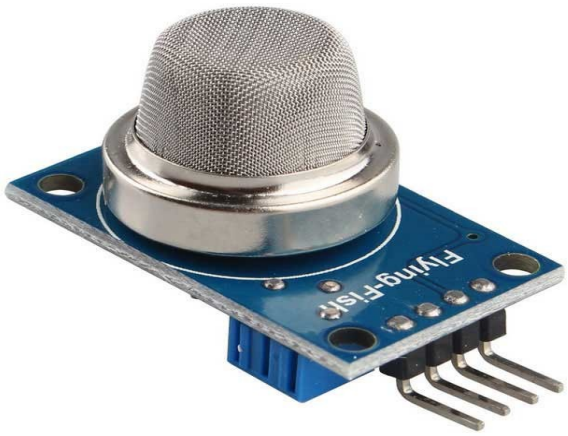
Arduino Uno



Arduino Uno is an open-source microcontroller developt by arduino.c.c. It consists of 6 analog and 14 digital pins. There are also pins for 5V , 3.3V and ground. The voltage source can be applied two ways : through DC power jack and USB port. A 9V batery could power this microcontroller , cause i powerd it through the USB port using the raspberry pi , there was need for that. Uploading codes to arduino is done through the USB port , by simply pluggin it to the pc and opening Arduino Ide (or VS extension Platform IO which i highly recomend) which will automatically connect to arduino and by just pressing a button upload it. I wrote it in VS code with Platform IO extension what is same as Arduino ide style and gives suggestion writing the code , what was the biggest disadvantage of Arduino ide. The programming language is a mixture between C and C++ which has many examples on the internet and a big cummunity. The biggest advantage of the arduino when comparing it to other microcontrollers it is community and learning curve. So its great for beginners and prototyping !

Sensors

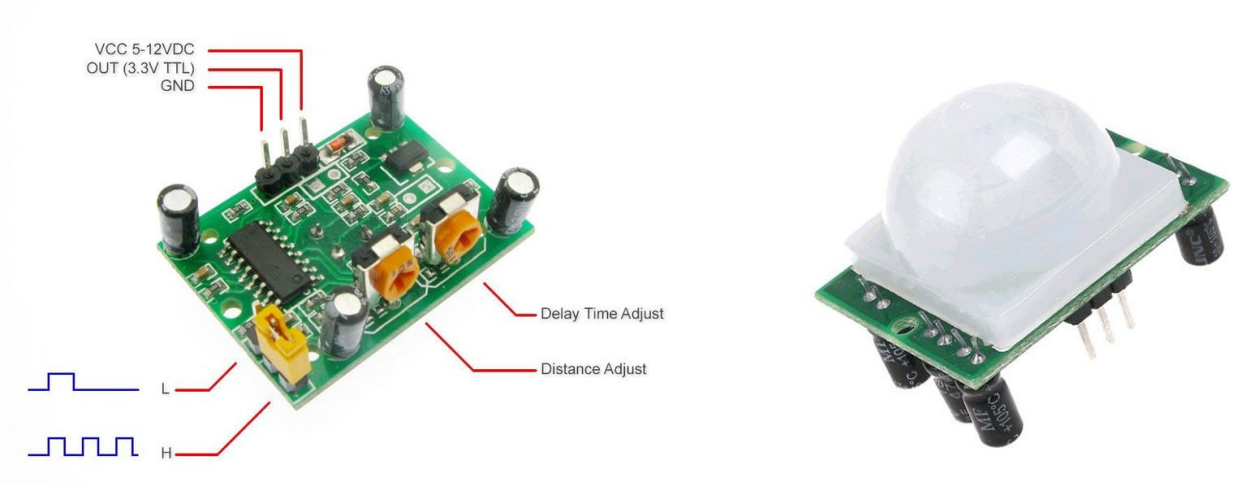
Gas Sensor[1]



That is the only hardware component which i didnt have but wanted to include in my project so i had to improvise , since we read just the analog output of the sensor i put a potentiometer instead to demonstrate this sensor. With changing the voltage i would demonstrate the change in smoke. As the smoke concentration rises so does the voltage , in other words 5V would be the maximum smoke and the residents would eventualy die , otherwise 0V what would mean there is no smoke. We connect this sensor to 5V and ground and measure the analog output with a pin.

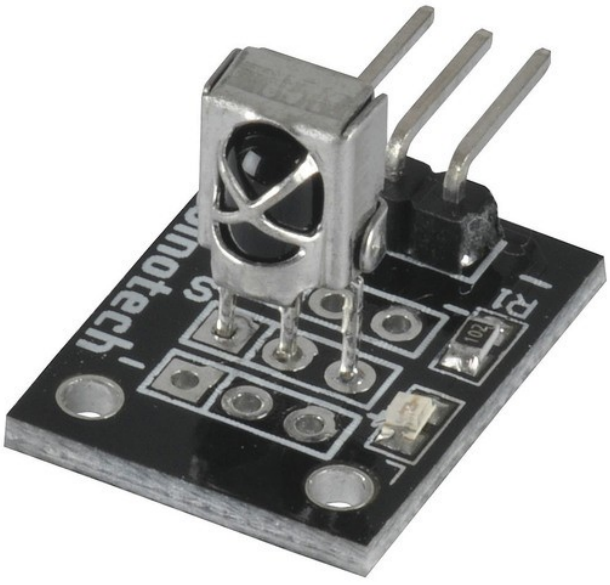
1 https://components101.com/articles/introduction-to-gas-sensors-types-working-and-applications

Pir Sensor[2]



PIR sensor stands for Passive Infrared sensor , so it measures infrared rediation in its enviroment and is used in most movement detection system. In the left figure we can clearly see that it has three pins (+V , ground , output) . The work principle is interestin since there are actually two sensors inside and a lot of things that affect this sensor. The main idea is that our body radiates infrared and when entering the range and area of the sensor it will measure it. It output high or low voltage so it is easy to handle.

IR reciver/remote module



IR reciver , are microchips with a photoresistor which filters just infrared signals. It is used in most remotes for Lcd displays and dcd players. It has a led which goes on when recieving signals . I has three pins , +V , ground and a pin which needs to be connected to PWM pins. That is because it will send the signals as hexadecimal numbers , when reciving it we need to assign the wanted option or button.
IR remote – is a IR transmitter , which by pressing a button will transmitt a IR signal.
The transmission is dependen on the Pulses send which is dependen on being either High or Low and on the lenght of them.

2https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work

Actuators

DC Servo motor

Are motor which specification is that they can rotate or push precisely. The movement can be controlled by the position or angle. They are used in many applications, perhaps robotics , machines … The difference to normal DC motors is that is goes through the servo mechanism. „It is a closed loop system where it uses positive feedback system to control motion and final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal“[3]. Similar as the IR reciver we need to connect it to a PWM pin .

16x2 LCD with I2C

The display has an LED backlight and can display up to 16 characters on two rows. One can see the square character fields as well as the pixels with which the characters are represented on closer inspection. The display shows the characters in white on a blue background and is intended to show text. Usually for controlling a LCD we would need 6 pins but with a PCF8574 chip we can use the I2C transmission protocol and use just 2 pins instead of 6. Those two pins we would connect to two analog pins , i connected it to A5 and A4.

Passive Buzzer

The functional principle of the passive buzzer is to make the air vibrate with PWM-generated audio signals. Depending on how high the frequency of the signal is, different tones are generated.
It has two pins and we connect it to one digital pin and to ground.

3 https://circuitdigest.com/article/servo-motor-basics

Arduion Code

First we have to import the libraries that we need for our project or task , in my case i used the following libraries . When programming in VS code i need to include the Arduino.h library , otherwise in the arduino ide . Next i defined the pins , other pins are automaticaly assign for its task as A4 and A5 for LCD I2C , for that i included the Wire library.

```
#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include "IRremote.h"
#include <Servo.h>
#include <TonePlayer.h>


#define GasPin A1
#define DirPin 2
#define ServPin 10
#define Soundpin 9
#define receiver 11
```

I will skip the variables and functions declaration since i will explan them through my code , so i will start with the setup function . As usuall i declered and assign most values and pins in my setup function , the connection_setup() will send „Start“ string to raspberry pi and wait to get a respons , after getting it it will print „Its connected“ and turn of the lcd off. Also i had the need to set the position of the servo , cause i had the problem when Alarm was on it was returning servo motor to orginal position . Like this i didnt find any problems according servo. The while function is same as delay i just use millis() for time managment.

```
void setup() {
  Serial.begin(9600);
  myservo.attach(ServPin);
  irrecv.enableIRIn();
  set_defaults();
  set_pins();
  set_lcd();
  lcd.print("PROGRAM STARTS..");
  connection_setup();
  time_now= millis();
  while(time_now+1000>millis())
  {}
  lcd.clear();
  lcd.noBacklight();
  myservo.write(pos);
}
```

```
void connection_setup(){

Serial.print("Start");

while(!connection)
{
  if (Serial.available() > 0) {
incoming = Serial.readString();
lcd.clear();
lcd.setCursor(0,0);
lcd.print(incoming);
if(incoming.equals("Connected"))
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Its connected");
  connection = true;
}
}
}
}
```

After the setup function comes the loop function which looks in my case similar it basically just cals two functions and checks has someone entered three times wrong pin in a row .

```
void loop() {

listen_pi();
listen_sensors();

 if (try_times== 3)
 {
    Serial.print("Times");
    sound_on();
 }
}
```

The listen function reads the data send from pi to arduino , for reading strings from serial port i used the readString() , we can just turn on/ off the alarm

```
void listen_pi()
{
  if (Serial.available() > 0)
  {
    incoming= Serial.readString();
    if(incoming.equals("Alarm"))
    {
      alarm = true;
    }
    else if(incoming.equals("AlarmOff"))
    {
      alarm= false;
    }
  }
}
```

In the listen_sensors we will check three sensors : gas sensor , IR reciever and movement sensor. For a normal gas sensor we would change the second and third parameter in the map function to 300 and 750. If one of the if statments are true the sound_on() function will be called , which will turn on the buzzer and be at that state until Raspi sends Soundoff string.

```
void listen_sensors()
{
  analog_read = analogRead(GasPin);

  analog_read = map(analog_read,0,1023,0,100);

  if(analog_read > 60)
  {
    Serial.print("Vatra");
    sound_on();
  }

  digital_read= digitalRead(DirPin);
  if((digital_read==HIGH)&&(alarm))
  {
    Serial.print("Lopov");
    sound_on();
  }
    if (irrecv.decode(&results))
  {
    readIR();
    irrecv.resume();
    print_screen();
  }

}
```

```
void sound_on(){
  tone1.tone (220);
  while(1) {
    if(Serial.available()>0)
    {
      incoming= Serial.readString();
      if(incoming.equals("SoundOff"))
      return;
    }

  }
tone1.noTone ();
}
```

After reciving an ir signal from the remote i will assign it to order char array and with a ifstatment do next steps. turn_on_lcd() will setbacklight on / off and delete the pin entered so if something went wrong we can delete it. check_result() send the current pin to the raspberry pi and opns door if true. add_to_num() function takes the char and adds to the current pin.

```
void readIR()
{
  switch(results.value)
  {
  case 0xFFA25D: strcpy(order, "Power"); break;
  case 0xFF02FD: strcpy(order, "Check"); break;
  case 0xFF6897: strcpy(order, "0");     break;
  case 0xFF30CF: strcpy(order, "1");     break;
  case 0xFF18E7: strcpy(order, "2");     break;
  case 0xFF7A85: strcpy(order, "3");     break;
  case 0xFF10EF: strcpy(order, "4");     break;
  case 0xFF38C7: strcpy(order, "5");     break;
  case 0xFF5AA5: strcpy(order, "6");     break;
  case 0xFF42BD: strcpy(order, "7");     break;
  case 0xFF4AB5: strcpy(order, "8");     break;
  case 0xFF52AD: strcpy(order, "9");     break;

  default:
    strcpy(order , "False");
  }

if(strcmp(order,"False") == 0)
  return;

do_order();
}
```

```
void do_order()
{
  if(strcmp(order,"Power") == 0)
  {
  turn_on_lcd();
  }
  else if((strcmp(order,"Check") == 0)&&(lcd_ligth == true))
  {
  check_result();
  }
  else if (lcd_ligth == true)
  {
   add_to_num();
  }
}
```

```
 void check_result()
 {
 Serial.print("Check");
 delay(10);
 Serial.print(strnumber);
 while (1) {
 if (Serial.available() > 0) {
 incoming = Serial.readString();
 if(incoming.equals("Right"))
 {
   authorization = true;
   open_door();
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("YOU ARE RIGHT");
   time_now = millis();
   try_times= 0;
   return;
 }
 else if(incoming.equals("False"))
 {
   strcpy(strnumber, "Wrong");
   try_times++;
   return;
 }
 }
 }
 }
```

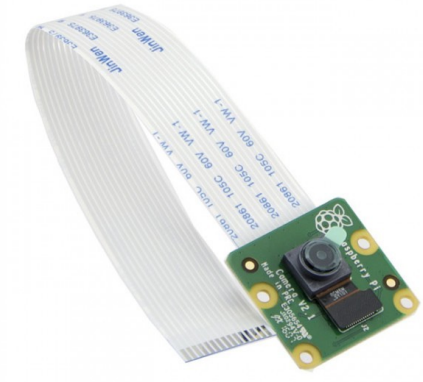
```
void add_to_num(){
if(num_position == 16)
return;

strcat(strnumber, order);
num_position++;
}
```

```
 void turn_on_lcd(){
 if(lcd_ligth == false)
 {
   lcd.backlight();
   lcd_ligth = true;
 }
 else
 {
 lcd.clear();
 lcd.noBacklight();
 strcpy(strnumber, "");
 lcd_ligth = false;
 }
 }
```

Raspberry Pi 3B+[4]

This is a raspberry pi 3B+ which i used ffor my project . Because i didnt use the pins i talk about the pin diagram. The only two things that are phisically connected with the raspberry pi are arduino and the camera module. Raspberry pi is a single board computer , which runs mostly with the rasbian OS (we can install varius OS on it). Rasbian is a unix based OS.
Some specifications are:

- 1GB LPDDR2 SDRAM
- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera (Next to the HDMI port)
- DSI display port for connecting a Raspberry Pi touchscreen display



I used a similar camera module as that in the image. It has a 5 megapixel camera. It can take photos up to 1080p and make video of 720p. The installation is pretty easy we just need to enable the camera options in the pi settings. Also for the usage we should have the comand line tools raspistill and raspivid installed. The installation are mostly easy on linux when having a knowledge of the comand line.

4 https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Code

My code consists of a c file and a header file. I could put everything in one c file but it would look messy and unprofessionell. With putting function declarations , structs and inlcude paths in the header we can have a better overview of the code.

First i will start to open or setting up my ports , variables and data bases. That assure us that everything should work fine. First we have a connection to the arduino . In that function i wrote a protocol between arduino and raspberry , when finishing that function i know that this port works. Then i check the server creation , works the socket , is it listening and so on. If server csocket works and we are listening „Great“ ! Then i check the databas and lastly i set the default values of my variables.

```
int main()
{

open_Arduino();
set_server();
set_database();
setdefaults();
```

After i know that ervery part works i can start running the real routine. Because i need the raspberry listening on two ports using threads is a good way to run my code parallel. Multithreading is a way of parallel programming . First we declare the threads , in my exmple thread for arduino and server. The we create them , first parameter is the thread adress, second the attributes ,third the function and lastly the argument. I just needed two parameters. And then with the pthread_join() we start them.

```
pthread_t threadarduino ,threadserver ;
int tr2 ,tr1;

tr1 = pthread_create(&threadarduino, NULL, listenArduino , NULL);
    if(tr1)
    {
        fprintf(stderr,"Error - pthread_create() return code: %d\n",tr1);
        exit(EXIT_FAILURE);
    }
tr2 = pthread_create(&threadserver, NULL, listenClient, NULL);
    if(tr2)
    {
        fprintf(stderr,"Error - pthread_create() return code: %d\n",tr2);
        exit(EXIT_FAILURE);
    }


pthread_join(threadarduino, NULL);
pthread_join(threadserver, NULL);
```

In the listenArduino function we are reading data send from arduino. Since the biggest char array that i send from the arduino is 5 bytes long i think that 10 elements are long enough . The password charr array needs to store 10 characters what is 11 bytes because we have \0 at the end. I my code the pin has to be 10 numbers long but we could also use a more dynamicaly aproach ,where the user could create one between 0-16 digit long but i think if we have a limit like 5 digits its more secure. num_bytes integer checks is the reading function working. We need to delete the value of BuffArduino everytime beacause then it could store from previuos reading.

```
void * listenArduino()
{

char BuffArduino[10];
char password[20];
int num_bytes;
    while(1)
    {
        bzero(BuffArduino ,sizeof(BuffArduino));
        num_bytes = read(serial_port, &BuffArduino, sizeof(BuffArduino));
```

The i will check the data after the redaing most instruction for fire or to much wrong entries will change the mode and if user turn on GUI inform him. The two functions which do something are function to check pin and if movement sensor has detected something. For the check pin case i am reading the buffer until i put 10 characters inside the password array. I need to know the reat characters of the first read buffer cause sometimes it can read like „Check123“ and we need to use this numbers.Then we add this numbers and from the new buffer. Lastly it will take the pin and check it if its right write it to raduino and insert new data into login database , otherwise send wrong to arduino. The movement sensor case will make a thread and make pictures with the camera module. Also i will change the alarm mode into ‚L‘ for robber , ‚S‘ stands for safe.

```
else if(strncmp(BuffArduino , "Check",5)==0)
{
    bzero(password, sizeof(password));
    int i = 0;
    int rest = strlen(BuffArduino)-5;
    for(i = 0;i<rest;i++)
        password[i]= BuffArduino[5+i];
    bzero(BuffArduino, sizeof(BuffArduino));
    while(rest<10)
    {
        read(serial_port, &BuffArduino, sizeof(BuffArduino));
        int s = strlen(BuffArduino);
        for(int j = 0; j<s;j++)
        {
            password[rest]= BuffArduino[j];
            rest++;
            if(rest == 10)
            break;
        }
    }
    if(Checkentrance(password)==1)
    {
        write(serial_port,"Right",strlen("Right"));
        InsertLogin();
    }
    else
    {
        write(serial_port,"False",strlen("False"));
    }
}
```

```
if(strncmp(BuffArduino , "Lopov",5)==0)
{
    alarmReason = 'L';
    printf("Lopov\n");
    pthread_t camerathread;
    int c = pthread_create(&camerathread, NULL, camera, NULL);
    if(c)
    {
        fprintf(stderr,"Error - pthread_create() return code: %d\n",c);
        exit(EXIT_FAILURE);
    }
    pthread_join(camerathread, NULL);
}
```

```
void InsertLogin()
{

time_t t = time(NULL);
struct tm *tm = localtime(&t);
char s[64];
assert(strftime(s, sizeof(s), "%c", tm));

sprintf(sql, "INSERT INTO EntLogin(Username, Time) VALUES( '%s' , '%s');",gazda.Name , s);

rc = sqlite3_exec(db, sql, 0, 0, &err_msg);

    if (rc != SQLITE_OK )
    {

        fprintf(stderr, "SQL error: %s\n", err_msg);

        sqlite3_free(err_msg);

    }
    else
    {
        fprintf(stdout , "Soucessfully inserted into login\n");
    }
}
```

This is the function which makes 60 pictures in around one minute. System() function writes to a comand line, in the sys array i insert the comand by the sprintf() function i can insert integer in char array similar as the print function. After each execution i take a delay of half second.

```
void * camera()
{
char sys[200];

    for (int j = 0; j < 60 ;j++)
    {
        sprintf(sys, "raspistill -o /home/pi/Desktop/Project/camera/image%d.jpg -t 10 -n -ex auto -awb auto -w 800 -h 600", j);
        system(sys);
        sleep(500);
    }

printf("Pictures token\n");

}
```

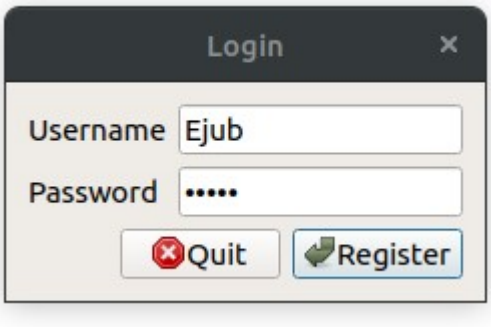
The next function is the listenClient() which is the second thread . It consist from one infinte while loop which will have two while loop iside. The first is after the connect function which will iterate until the client wants to check the password login. That assures us that we could not comunicate until the password is right. If the password is right it will also send the client some information as is the alarm on or is the house safe or has it some problems. The Checkclient functions selscts the user with the same name as first parameter and then the password. If both are same it will return 1 , otherwise it will return 0. If it is 1 we will exit while loop and enter the second while loop , otherwise we will enter a new time and wait for password.

```
while(1)
{
    char bufferClient[MAX];

    len = sizeof(cli);

    connfd = accept(sockfd, (SA*)&cli, &len);

    if (connfd < 0)
    {
        printf("server acccept failed...\n");
        exit(0);
    }
    else
    {
        while(1)
        {
            bzero(bufferClient, MAX);

            read(connfd, bufferClient, sizeof(bufferClient));
            if (strncmp("Check Pass", bufferClient, 10 ) == 0)
            {
```

```
while(bufferClient[n]!=' ')
{
    u[n] = bufferClient[n];
    n++;
}

n++;
int j= 0;

while((bufferClient[n]!='\0')&&(bufferClient[n]!='\n'))
{
    u[j] = bufferClient[n];
    n++;
    j++;
}
if(Checklogin(u,p))
{
updateClient();
break;
}
else
{
continue;
}
```

The second loop is used for comunication between raspberry and GUI. There are five instructions , moovement sensor on or off , turn down alarm , update and exit (which will exit the connection). Most of it is simple and we already locked at similar code for arduinoListen. This is the main idea for the raspberry code , so i wont go to much in detail. The raspberry and arduino can work independent and dont need the client to be connected. The client is just there to change the mode.

GUI



For the GUI i used QT. „Qt is a cross-platform application development framework for desktop, embedded and mobile. Supported Platforms include Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS and others. Qt is *not* a programming language on its own. It is a framework written in C++“[5].I would recomend it for everyone. I had some experience from a previous project and it is indeed a powerfool tool for GUI programming. I find it much better than perhaps Swing. The first dialog is login dialog. We can insert the username and password. Then a new dialog should pop up and we could change the settings.

Conclusion

It was a very intresting project. I like that i learned a lot of things in electronic and programming as database , network programming , linked lists … The project has a good base and should be improved (What i am planing). Adding a better GUI with few more options would be great. Also i could add a few components and lines of code and make a smart home system. It was a nice experience ,sometimes stressing with some mistakes but everything got solved. I found the difference between theoretical nad practical subjects of this subject , in theoretical evrithing is easy but in practical there sis always some problem. All in all it was a nice experience and i enjoyed it.

5 https://wiki.qt.io/About_Qt

References:

https://www.arduino.cc/en/Tutorial/HomePage?from=Main.Tutorials
https://www.raspberrypi.org/documentation/raspbian/applications/camera.md
https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md
https://aticleworld.com/socket-programming-in-c-using-tcpip/
https://www.binarytides.com/socket-programming-c-linux-tutorial/
http://zetcode.com/db/sqlitec/
https://computing.llnl.gov/tutorials/pthreads/
http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
https://www.youtube.com/watch?v=QbFM0YroaF0
https://magpi.raspberrypi.org/articles/program-arduino-uno-raspberry-pi
https://ftp.gnu.org/old-gnu/Manuals/glibc-2.2.3/html_chapter/libc_32.html
https://www.qt.io/