SMART HOME PROJECT

TEAM 6

Name	ID	section
Omar Ahmed Abbas	**10047E**	**Software Team** - Computer vision - Add users to the database - Connecting TK windows together - Connecting Arduino to python
Zeyad Mahmoud Abdalrahman	**100142E**	**Hardware Team** - Altium design files - Circuit fabrication
Yehia Mahmoud Abdelazim	**10087E**	**Hardware Team** - Altium design files - Proteus design - Circuit fabrication
Reem Ibrahim Desouky	**100157E**	**Microcontroller Team** - **Upload code to ATmega** - **Tinkercad simulation** - **Arduino programming**
Menan Mohamed Mostafa	**100135E**	**Software Team** - Design GUI windows with Tkinter - Change password option and save it in text file in the device

Mind Cloud

Ultimate Innovation

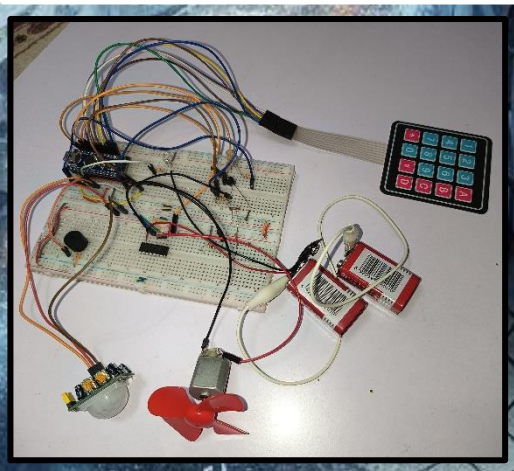
Report Outlines

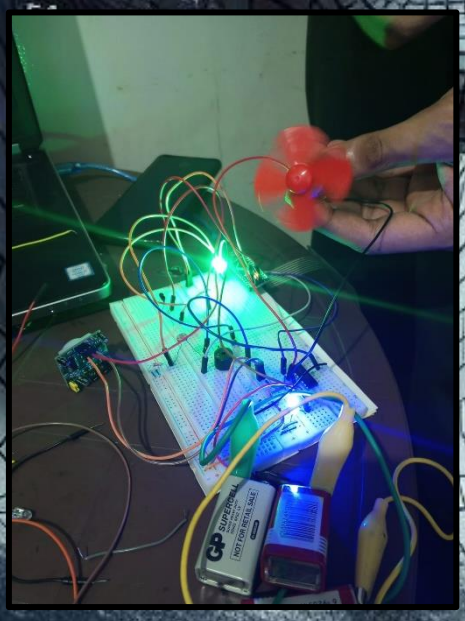
Part one : Software

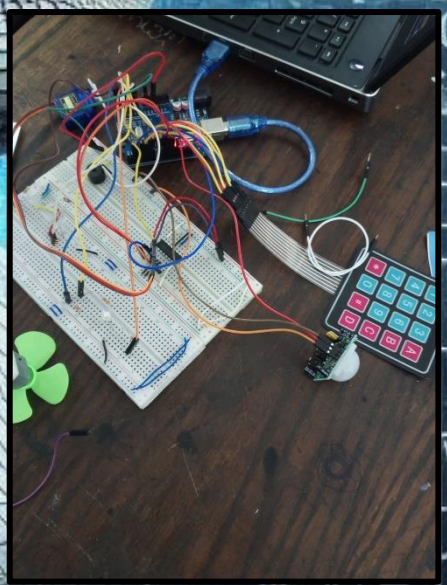
- Computer vision
- GUI design
- How to upload code on ATmega8A
- ATmeaga8A code
- All python code

Part two : Hardware

- Components
- Tinker cad (Arduino)
- Proteus
  1- Schematic
- Altium files
  1- Library
  2- Schematic
  3- Fabricated PCB
- Fabricated circuit







Computer vision

The first step in creating a smart house is implementing a Face Recognition system. We began by researching how to build a Face Recognition system, tested it with our photos, and it worked correctly.

Step 1 - used libraires in the code

1] OpenCV : It helps with image and video processing, including tasks like object detection, face detection, and manipulating images

2] face_recognition : This library is built on top of the powerful dlib library and simplifies tasks related to face recognition and manipulation. It's widely used because it abstracts complex face recognition tasks into easy-to-use functions.

3] OS : module is a built-in Python library that provides functions for interacting with the operating system. It allows for file and directory manipulation , we use it to get our database photos file

4] Numpy : NumPy is used for one specific purpose, finding the index of the minimum value in an array

All code (face Recognition only) can find in this link :

https://drive.google.com/file/d/1QrrNzC8wPyne1nEvhjuEbQac-n666QjI/view?usp=drivesdk

Step 2 - Face Recognition Code

1 This is the first part of the code that import all libraires used in the code and then defines the path to the folder where team photos are stored, and initializes an empty lists to store the face encodings of known team members, and names of the people corresponding to the face encodings

```
import cv2
import numpy as np
import face_recognition
import os

family_folder = "Family"
known_face_encodings = []
known_face_names = []
```

2 The second part of the code is responsible for processing each image in a specified folder to extract and store face encodings and corresponding name, These encodings are later used for comparing and recognizing faces in live video

```
for filename in os.listdir(family_folder):
    if ( filename.endswith(".jpg") or filename.endswith(".png") ):
        family_image = face_recognition.load_image_file(os.path.join(family_folder, filename))
        family_encoding = face_recognition.face_encodings(family_image)[0]
        known_face_encodings.append(family_encoding)
        known_face_names.append(os.path.splitext(filename)[0])
```

3 for the first line in third part, This line initializes a face detection classifier using a pre-trained model, For the second line opens a video capture stream from the default camera, and for the last line Applies the Haar cascade face detection to the captured frame. This function detects faces and returns a list of rectangles where faces are found

```
face_detect = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
vid = cv2.VideoCapture(0)

while True:
    ret, frame = vid.read()
    if not ret:
        break
    faces = face_detect.detectMultiScale(frame, 1.3, 5)
```

4 This part of the code performs several operations to detect faces in a video frame, draw rectangles around them, and recognize if the detected faces match any known faces using the face recognition library, The closest matching face is identified, and if it matches any known person, their name is displayed; otherwise, the face is labeled as "Unknown"

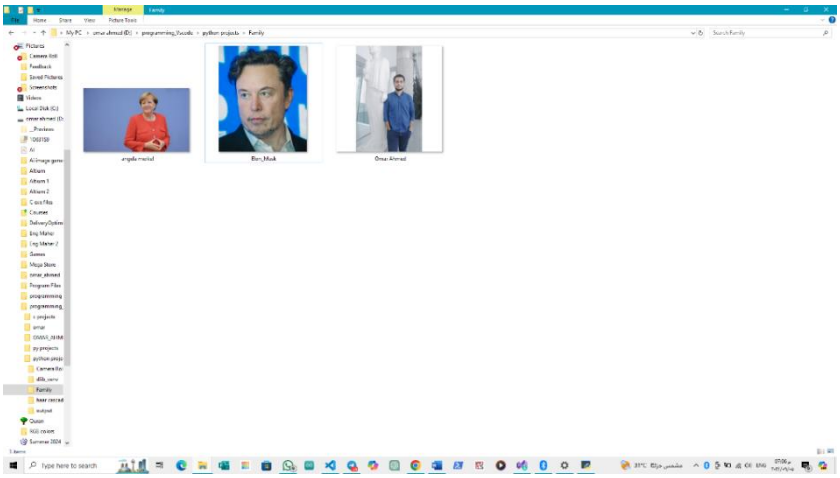
```
for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    face_only = frame[y:y + h, x:x + w]
    rgb_face = cv2.cvtColor(face_only, cv2.COLOR_BGR2RGB)
    face_encodings = face_recognition.face_encodings(rgb_face)
    if face_encodings:
        face_encoding = face_encodings[0]
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding, tolerance=0.6)
        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]
        else:
            name = "Unknown"
```

5 This part of the code is responsible for resizing the frame, adding rectangles and text to the frame, and displaying it in a window, Finally, the modified frame is displayed in a window for the user to see. This process runs continuously while the program is capturing video

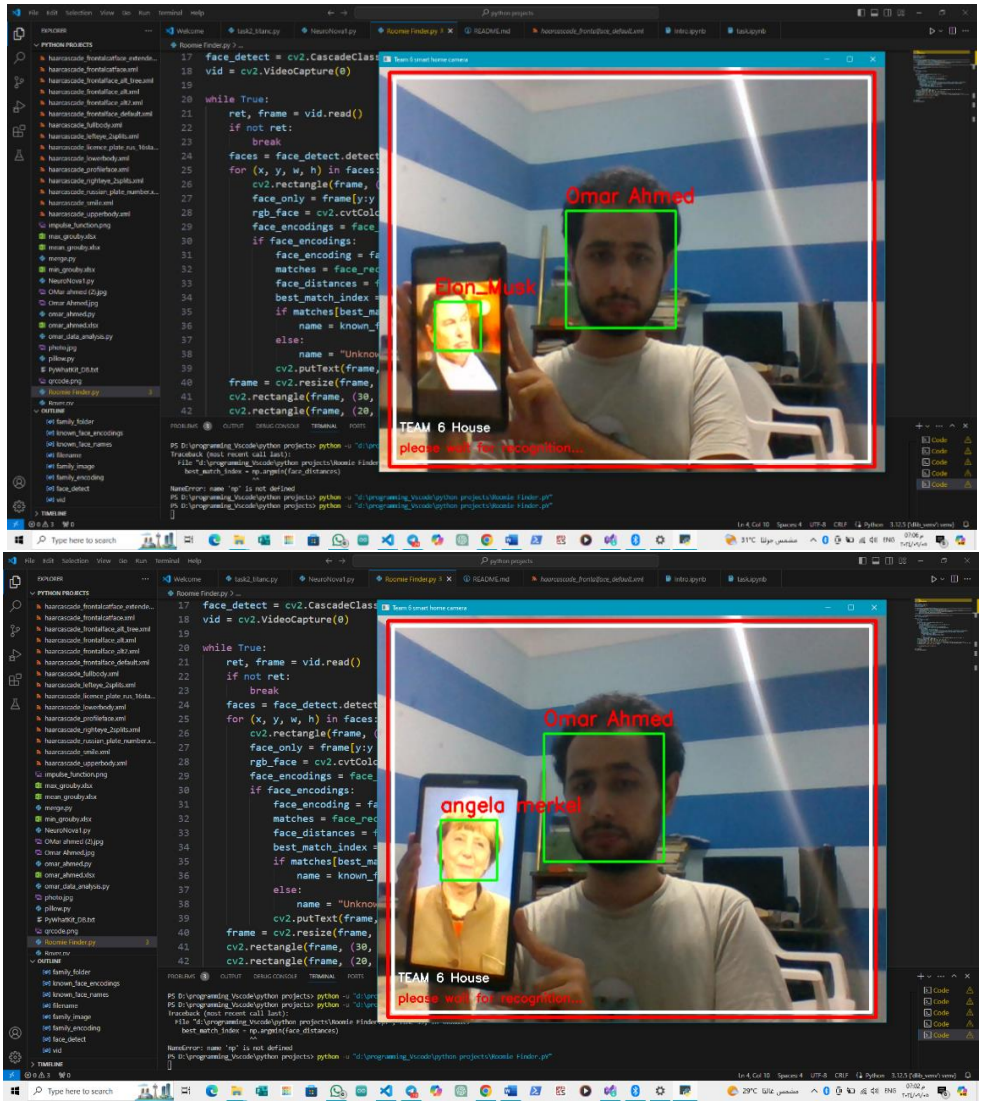
```
frame = cv2.resize(frame, (1000, 800))
cv2.rectangle(frame, (30, 780), (970, 20), (255, 255, 255), 5)
cv2.rectangle(frame, (20, 790), (980, 10), (0, 0, 255), 5)
cv2.putText(frame, "TEAM 6 House", (40, 720), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)
cv2.putText(frame, "please wait for recognition...", (40, 760), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
cv2.imshow("Team 6 smart home camera", frame)
```

Test for the code

1] database (photos folder , example data)



2] test (with different photos)



GUI Design

The second step is to design a GUI instead of the terminal , its propose to show status of the house, change password, enter password, open the camera , and it consider the house gate that with it you can control every thing

Step 1- used libraires in the code

1] tkinter : the standard Python library used for creating graphical user interfaces (GUIs). It provides a way to build desktop applications with elements like windows, buttons, labels, text boxes, and more

2] tkvideo : Python package that allows you to play video files (like MP4) in a Tkinter GUI. Tkinter, by default, does not have built-in support for playing videos, but tkVideo provides this functionality by enabling video playback directly within a Tkinter window.

Step 2- GUI code

1] first window :

This is the first window appear when the user run the code.

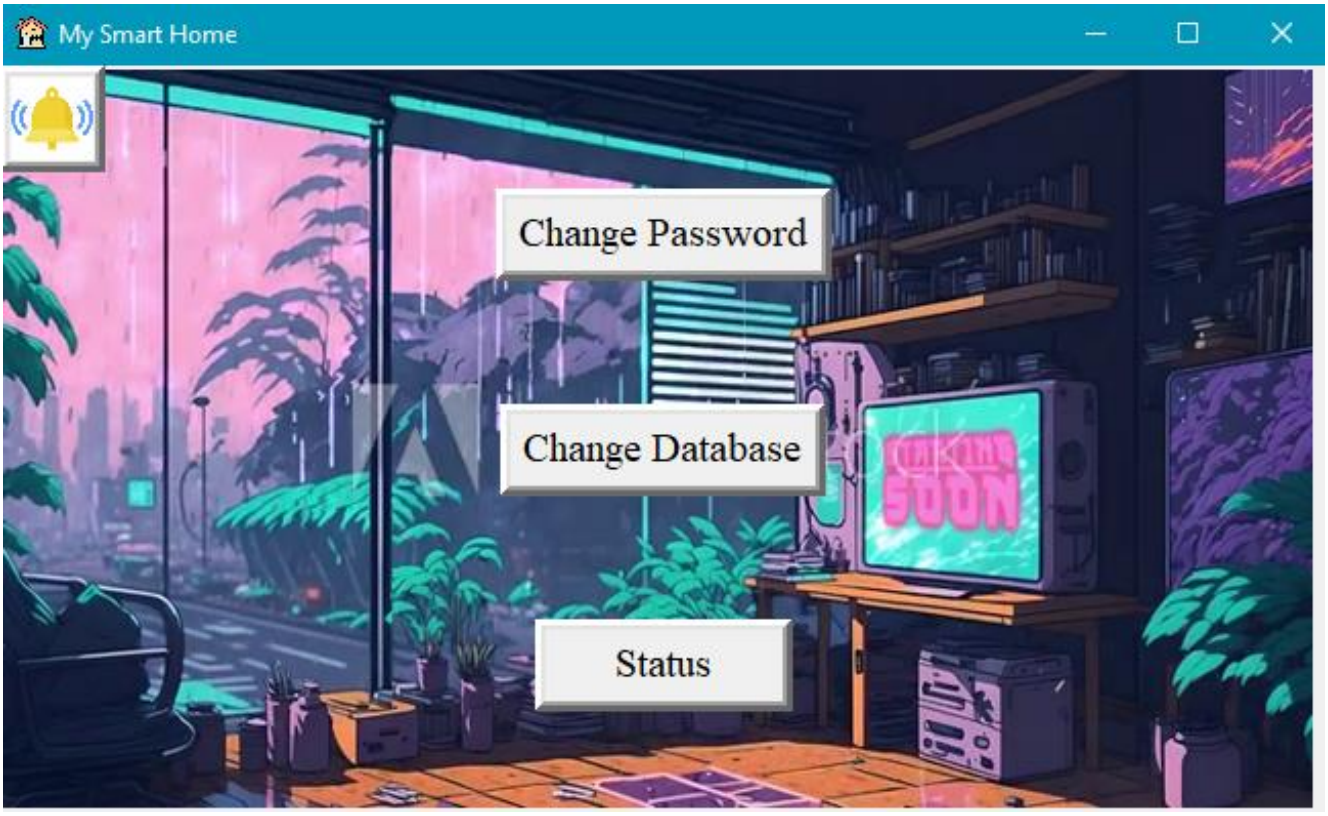
when the user click the Button, then the camera of laptop will open in few seconds and recognize the face of the user, so the button command is call the function that will open the camera and this is consider the first step to enter the house

```
main_window = Tk()
main_window.geometry("650x380")
main_window.title("House of Team 6")

logo = PhotoImage(file='SH2.png')
main_window.iconphoto(True, logo)

lbl = Label(main_window)
player = tkvideo("BG2.mp4", lbl, loop=1)
player.play()
lbl.place(x=0, y=0)

button1 = Button(text="Enter the house",
                 bd=5,
                 width=40,
                 height=20,
                 font=("Times New Roman",18),
                 command=open_camera)
button1.pack(pady=150)

main_window.mainloop()
```

2] second window :

After the camera open and determine whether the face is in the database or not, then it has two options , the first option is when the face in the database and camera recognized him/her, then it will open the second window that appear only if you are in the database , you can then change the password , change the database or read the status of the house, such as if there is person on the house, the temperature of the house, and other.

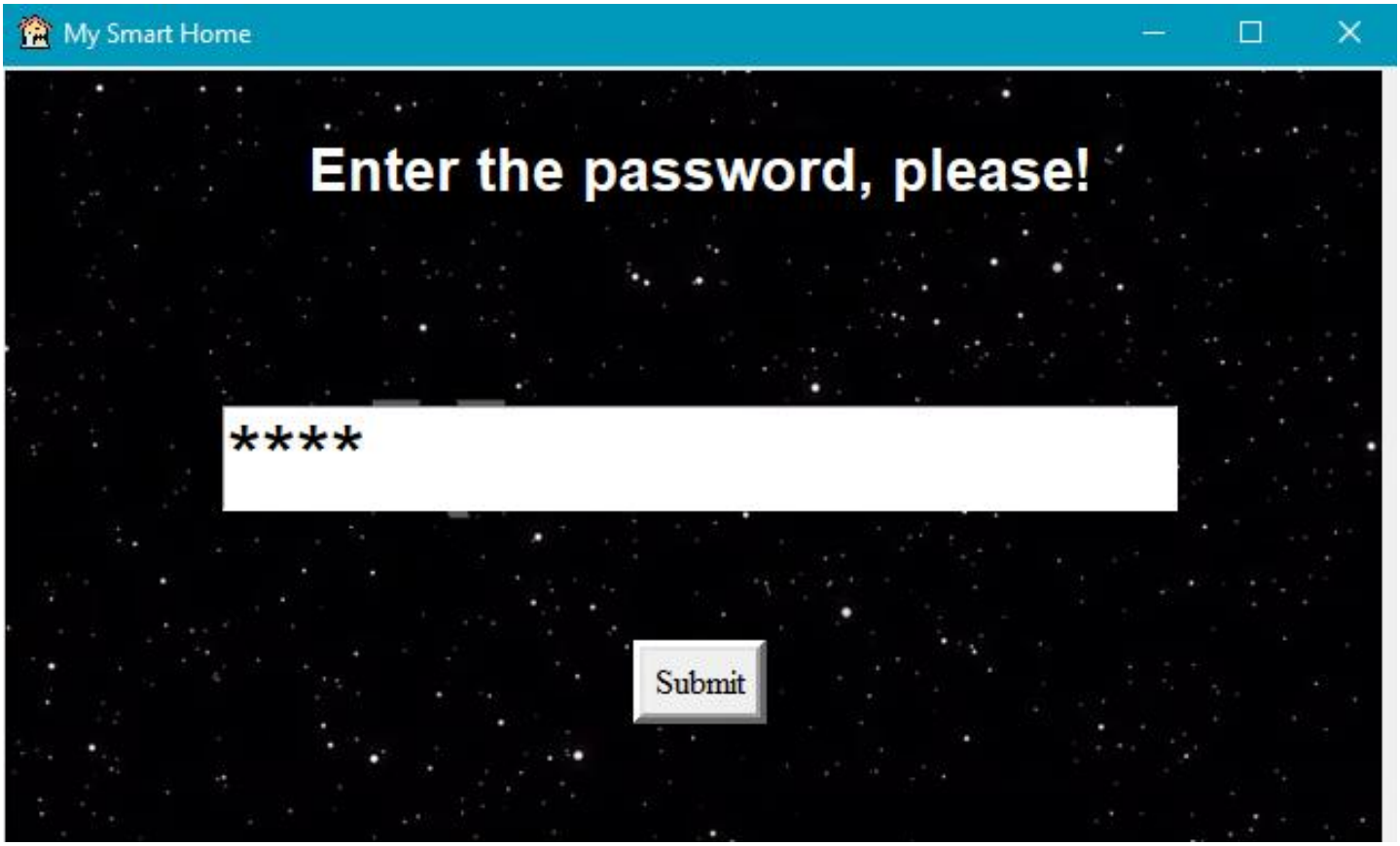
```python
from PIL import Image, ImageTk

window2 = Tk()
window2.geometry("650x380")
window2.title("My Smart Home")

logo = PhotoImage(file='SH2.png')
window2.iconphoto(True, logo)

lbl = Label(window2)
player = tkvideo("background.mp4", lbl, loop=1)
player.play()
lbl.place(x=0, y=0)

button1 = Button(window2,
                 text="Change Password",
                 font=("Times New Roman",15),
                 bd=5)
button1.pack(pady=60)
button2 = Button(window2,
                 text="Change Database",
                 font=("Times New Roman",15),
                 bd=5)
button2.pack()
button3 = Button(window2,
                 text="Status",
                 width=10,
                 font=("Times New Roman",15),
                 bd=5)
button3.pack(pady=60)
image = Image.open("noti2.png")
resized_image = image.resize((40, 40))
logo2 = ImageTk.PhotoImage(resized_image)
button4 = Button(window2,
                  image=logo2,
                  width=40,
                  height=40,
                  bd=5)
button4.place(x=0, y=0)
window2.mainloop()
```



3] third window :

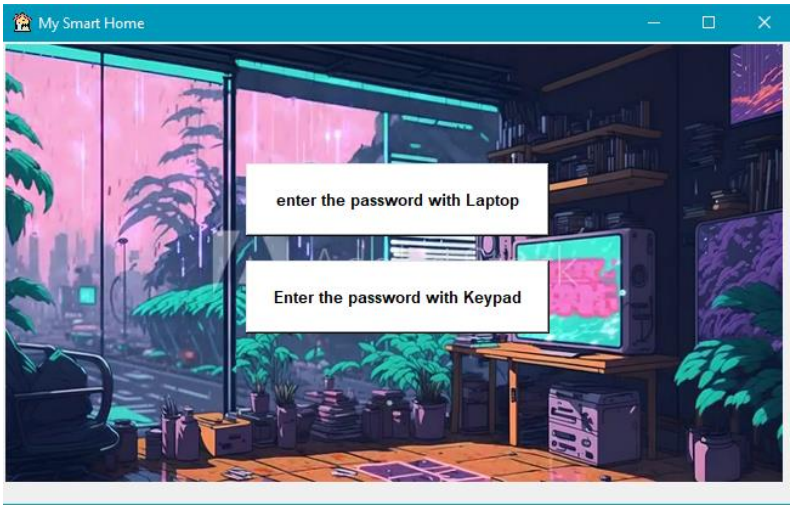
This is the second option that must appear when the face isn't in the data base, it mean that the user must enter the password correctly, so if the user enter it correctly then the second window will open (second window) , and if the user enter the wrong password then he must try again until the correct password or he will stay in the same window

```python
from tkinter import messagebox
from subprocess import call

def checkpass():
    enter = entry.get()
    if enter == "0123" :    # MUST make pass as global var
        window1.destroy()
        call(["python","app.py"])
    else :
        entry.delete(0,END)
        messagebox.showerror(title="ERROR!",message="INCORRECT PASSWORD")


window1 = Tk()
window1.geometry("650x380")
window1.title("My Smart Home")
logo = PhotoImage(file='SH2.png')
window1.iconphoto(True, logo)
lbl = Label(window1)
player = tkvideo("BG2.mp4", lbl, loop=1)
player.play()
lbl.place(x=0, y=0)
text = Label(window1,text="Enter the password, please!",
             font=("Arial",20,"bold"),
             fg="white",
             bg="black")
text.pack(pady=30)
entry = Entry(window1,
              width=20,
              font=("Arial",30),
              show="*" )
entry.pack(pady=60)
submit = Button(window1,
                text="Submit",
                font=("Times New Roman",12),
                bd=5,
                command=checkpass)
submit.pack()
window1.mainloop()
```



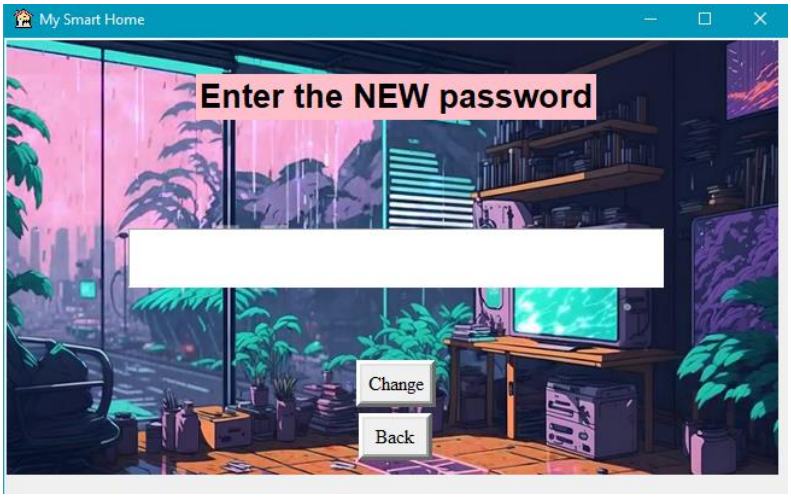
4] password enter method

You are free to choose which method you prefer to enter the password, by keypad or with laptop, this window appear when your face isn't in the database



5] Change password window

This window allows you to change the password and then save the change in TXT file to use it later



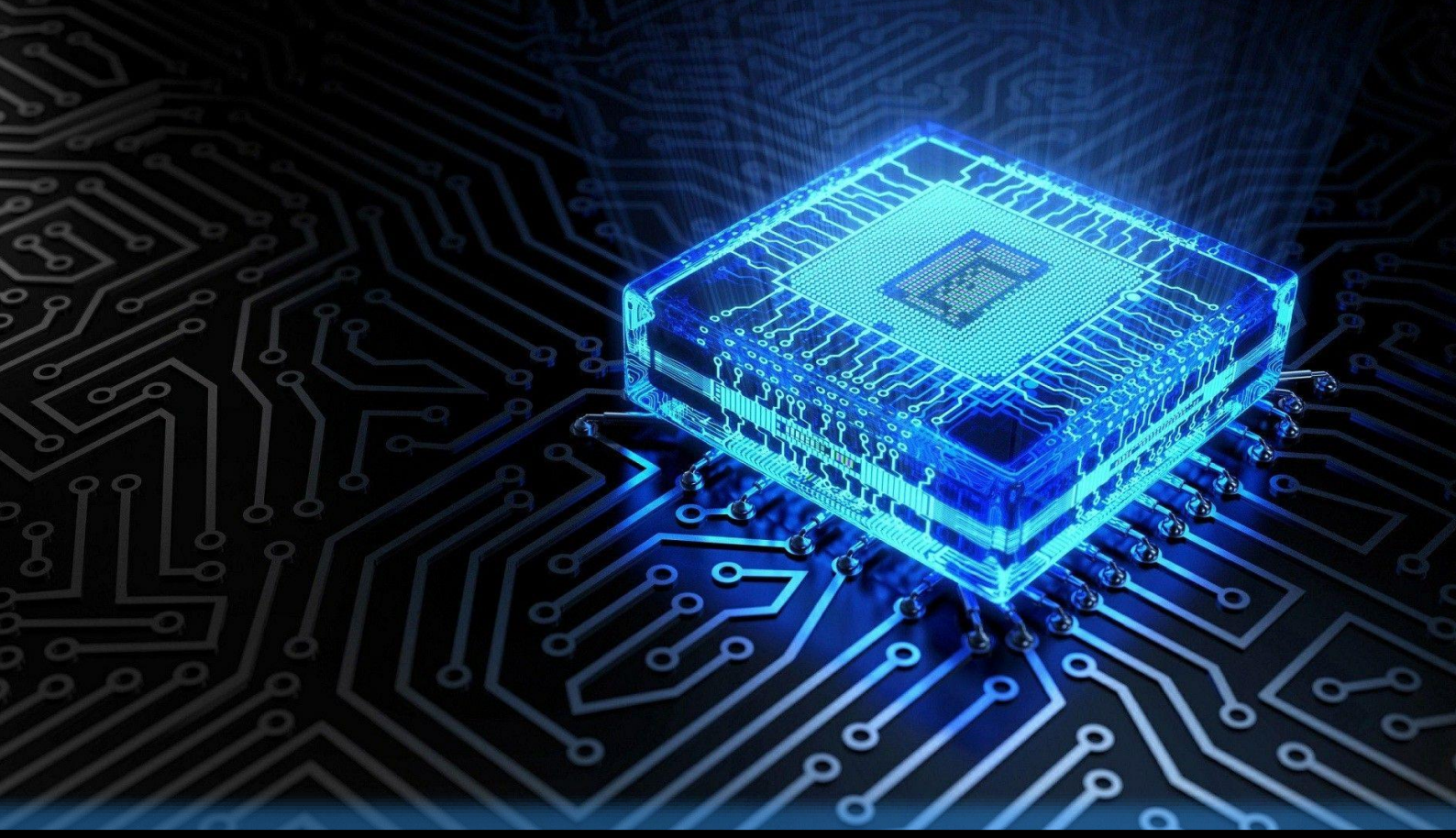
Videos for the GUI :

1] what will happen when change the password

https://drive.google.com/file/d/1VF3YDzoSHdKE5rzCDnHQ0UhsUXlXIfyP/view?usp=drivesdk

2] what will happen when enter wrong password

https://drive.google.com/file/d/1VWyPCaOIdev4ASeucPUipeIPeGXZjkvB/view?usp=drivesdk



How to upload the code on ATmega8A

At the very beginning of the final project phase, we deeply read and understood the required assignment according to this section, as we have to submit a detailed algorithm for the following :

1- Opening the door using a servo motor “whether by facial recognition or password entry”.

2- Measuring sensors values.

3- Take action if password is entered wrong

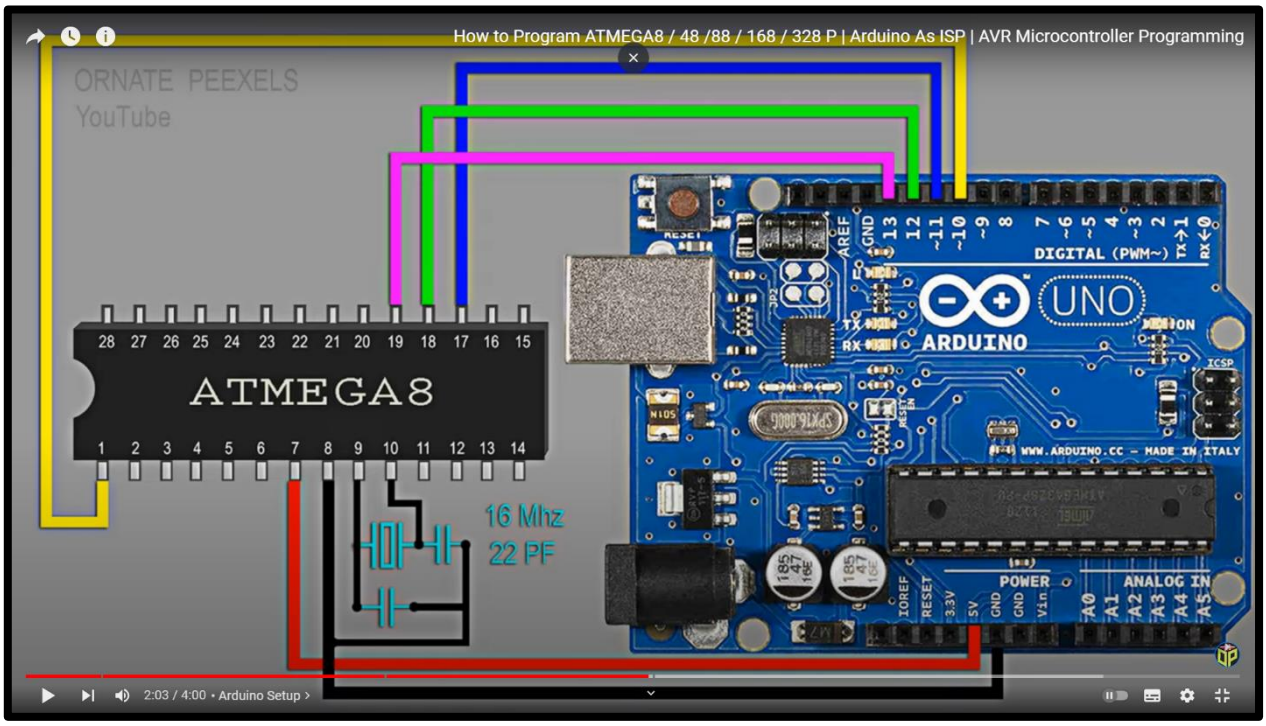
Therefore, we started working on the simulation on Tinkercad just to examine the code, and to determine the beginning of the designing phase for the hardware so that the hardware team can work on a specific, examined design. You can have a look at our simulation through this Smart Home (tinkercad.com)

Problems we encountered and steps to solve them

Firstly, we noticed that we have to work on ATMEGA 8A instead of a full Arduino kit which was challenging for us because of the way that we will upload the code on the ATMEGA and what language that we will be writing with; additionally, we have to work on specific sensors that we are not familiar with such as the NTC Thermistor Module and the LDR sensor. So, we searched for the new things and it turned to be easier than we thought as we found a very simple way to program the ATMEGA using an ISP, and it can

be done by an ordinary Arduino because the Arduino has an ISP within. Then, the NTC Module is available in stores, and the LDR sensor is the same as the ordinary photoresistor. In addition, we studied the ATMEGA pins, and we found out that its pin configuration is easy to deal with, and it is very similar to the Arduino's. Secondly, while working on the simulation on Tinkercad, we noticed that there was no ATMEGA and NTC modules in there. First, we were thinking of doing the simulation on Protus Software, but we won't be able to examine our code. So we decided that maybe we could work on Protus' simulation later, but for now we ought to work on the code. Hence, we used the ordinary Arduino kit and temperature sensors, and later we will adjust the code for the required.

Method of upload the code to ATmega8a using Arduino as a ISP programmer



Step 1 : open Arduino IDE

Step 2 : go to file from top bar and choose preferences

Step 3 : paste this link in URL entry :

https://mcudude.github.io/MiniCore/package_MCUdude_MiniCore_index.json

Step 4 : go to Tools from top bar and choose Minicore

Step 5 : from Minicore choose ATmega8

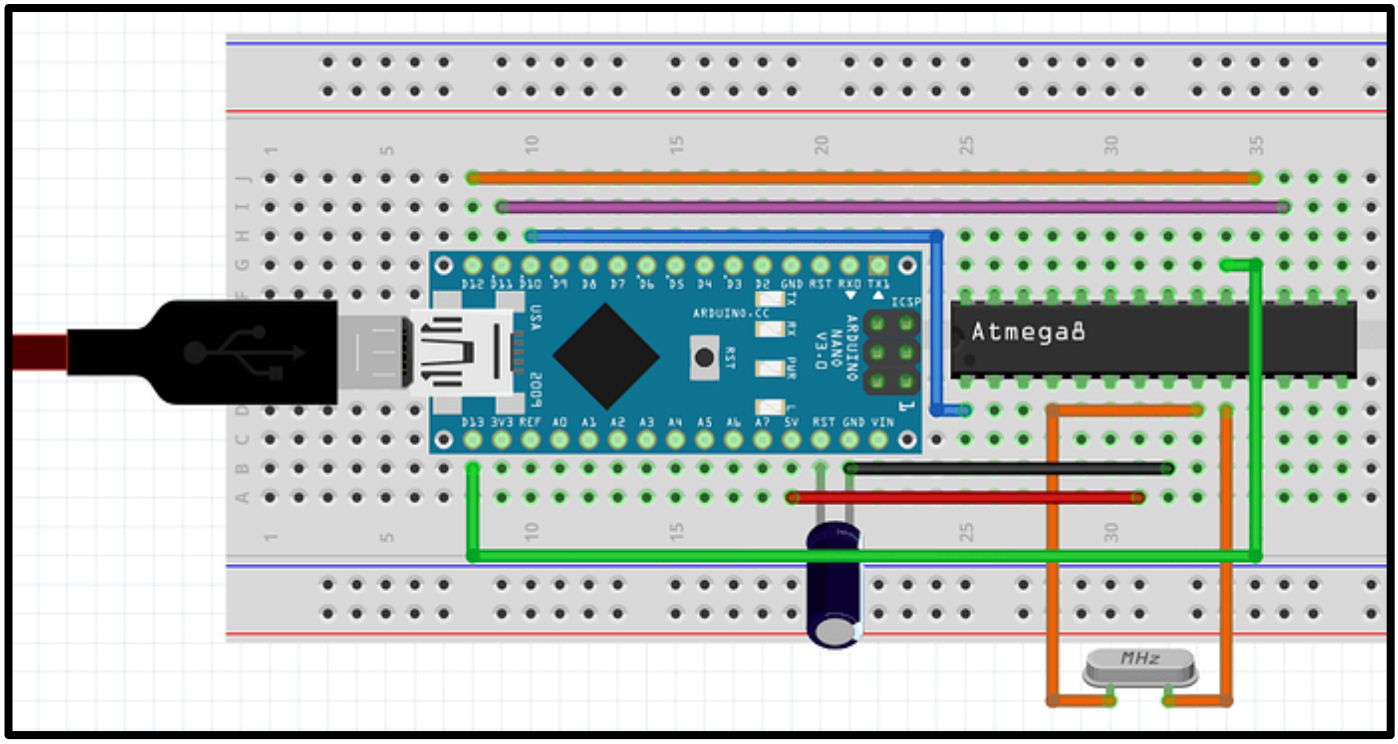
Step 6 : check from clock that it is External 16MHZ

Step 7 : choose programmer : Arduino as ISP

Step 8 : check BOD : BOD 2.7V

Step 9 : after write the code on IDE click on sketch

Step 10 : click on upload using programmer



Method of using USB port (laptop) with ATmega8A to full control :

A **USB to TTL UART uploader module** is a device that enables communication between a computer's USB port and devices using **TTL (Transistor-Transistor Logic)** serial communication via the **UART (Universal Asynchronous Receiver-Transmitter)** protocol. It's commonly used for programming or debugging microcontrollers, embedded systems, and other devices that use UART communication.

It provides a bridge between USB (computer) and UART-based devices (serial communication), enabling data exchange for debugging, logging, or communication purposes.

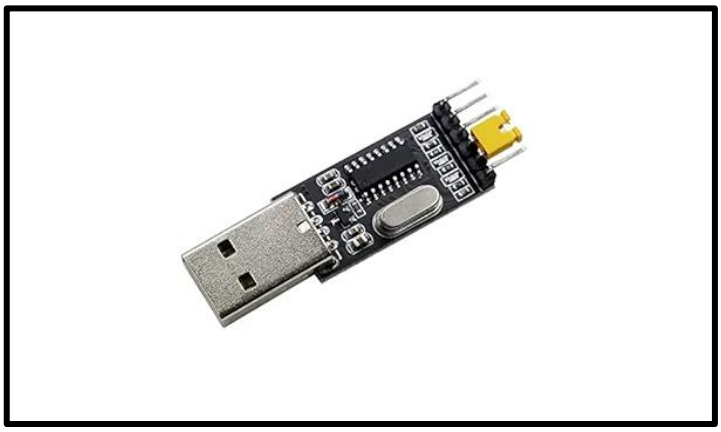
you can use a **USB to TTL UART module** to connect **PySerial** with an **ATmega8** microcontroller.

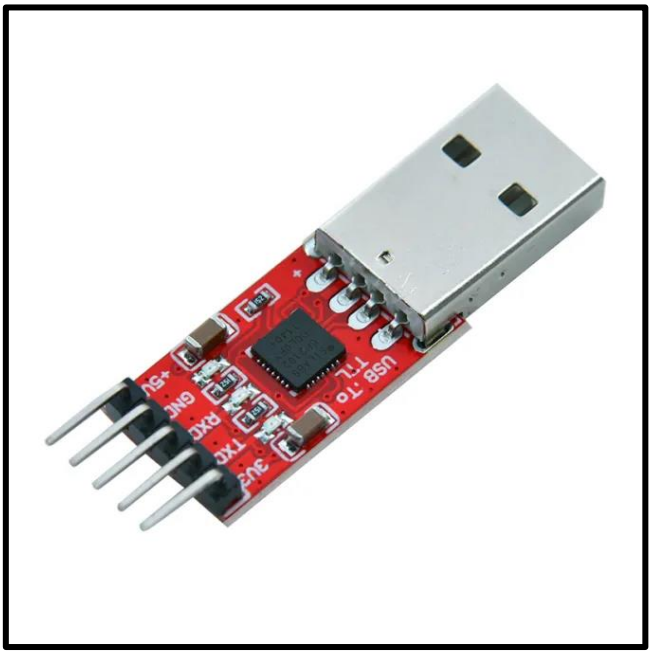
Wiring Connections :

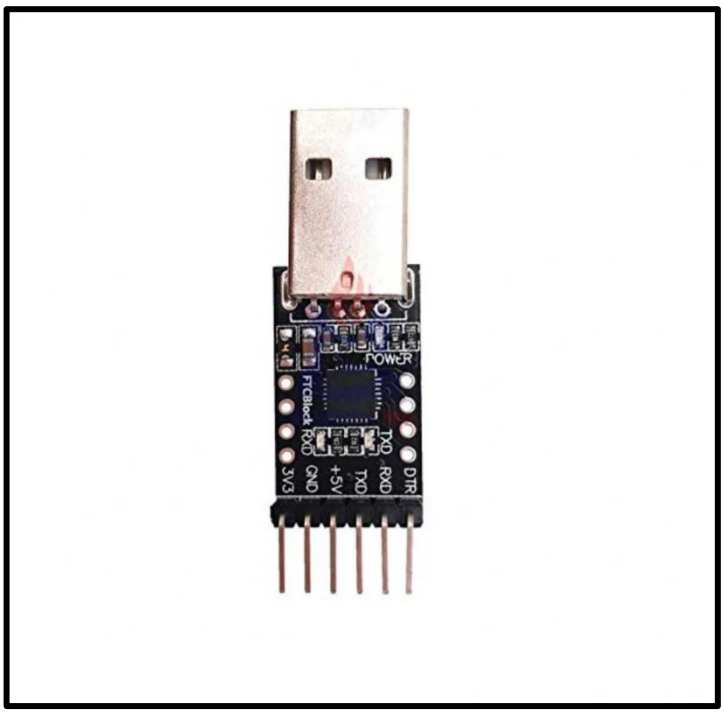
1- **TX (Transmit) of the USB to TTL Module → RX (Receive) of ATmega8**.
2- **RX (Receive) of the USB to TTL Module → TX (Transmit) of ATmega8**.

3- **GND (Ground) of the USB to TTL Module → GND of ATmega8**

4- **VCC of the Module**: Depending on the module, you may provide power to the ATmega8 using the USB to TTL module's VCC pin. Ensure the voltage levels (3.3V or 5V) are compatible with your ATmega8.





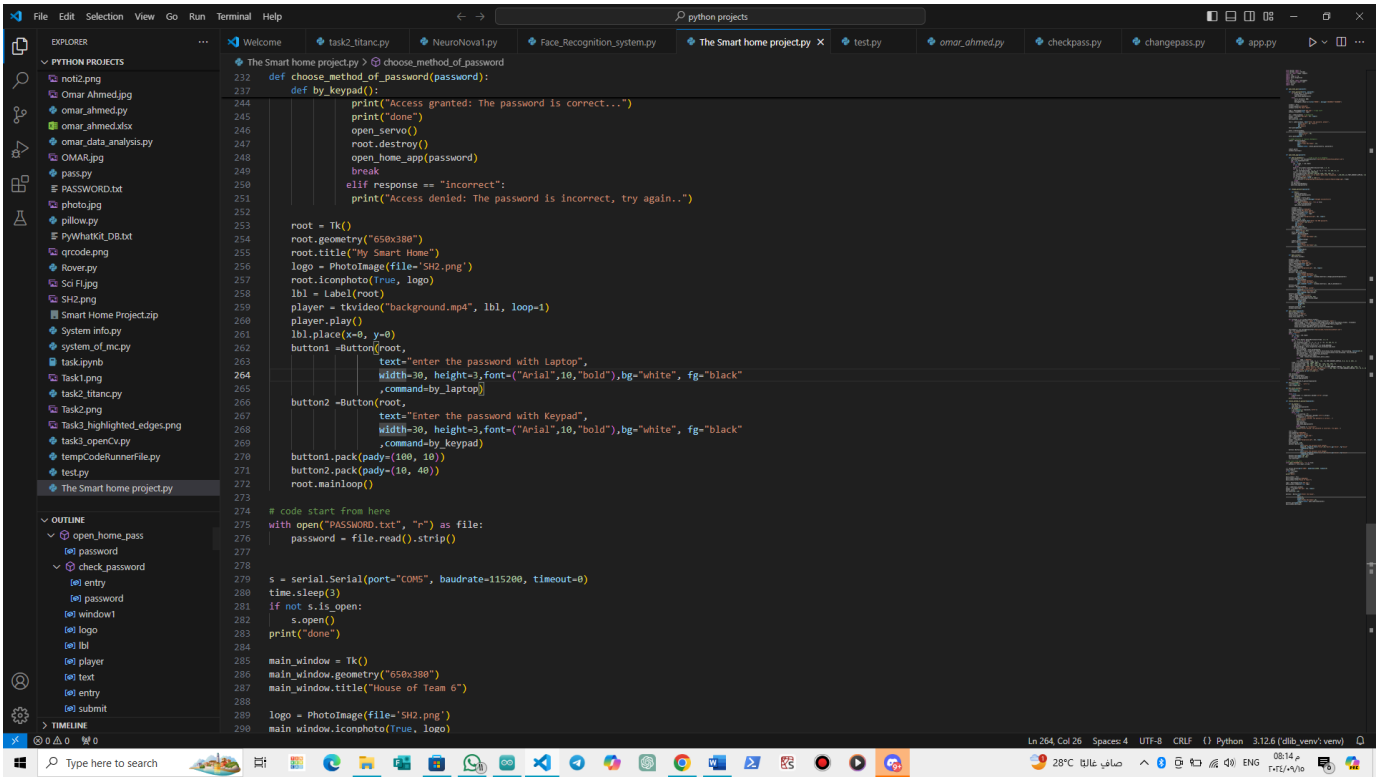


Microcontroller (ATmega8a) Code

Through the past 3 days, we have been working on the Tinkercad simulation, and by now we succeeded in completing both the simulation and the code. Code can be found in the code section through the Tinkercad link (**Smart Home (tinkercad.com)**) Additionally, the code performs all the required functions, but we are still developing and adding extra features to it

we had lots of problems handling the code altogether; for example, we had a problem on allowing someone new to enter the house while someone was already inside. That was challenging as we didn't want the system to shut up while wanting someone outside to check the facial footprint and/or password, so we searched and debugged the code many times until reached the proper way to solve this problem. Another problem that we faced was how to receive the commands from the GUI, and how to check the password while it was stored in a file that can be accessed by the computer vision's code only. Therefore, we did a serial connection between both codes that we was able to send/receive files to/from each other easily. Finally, we finished the code except for one point which is changing the temperature sensor from TMP36 to NTC Thermistor Module, and we will work on that once we are uploading the code on the microcontroller. Please notice that we used TMP36 in the simulation because there was no NTC Thermistor Module in Tinckercad as well as the ATMEGA

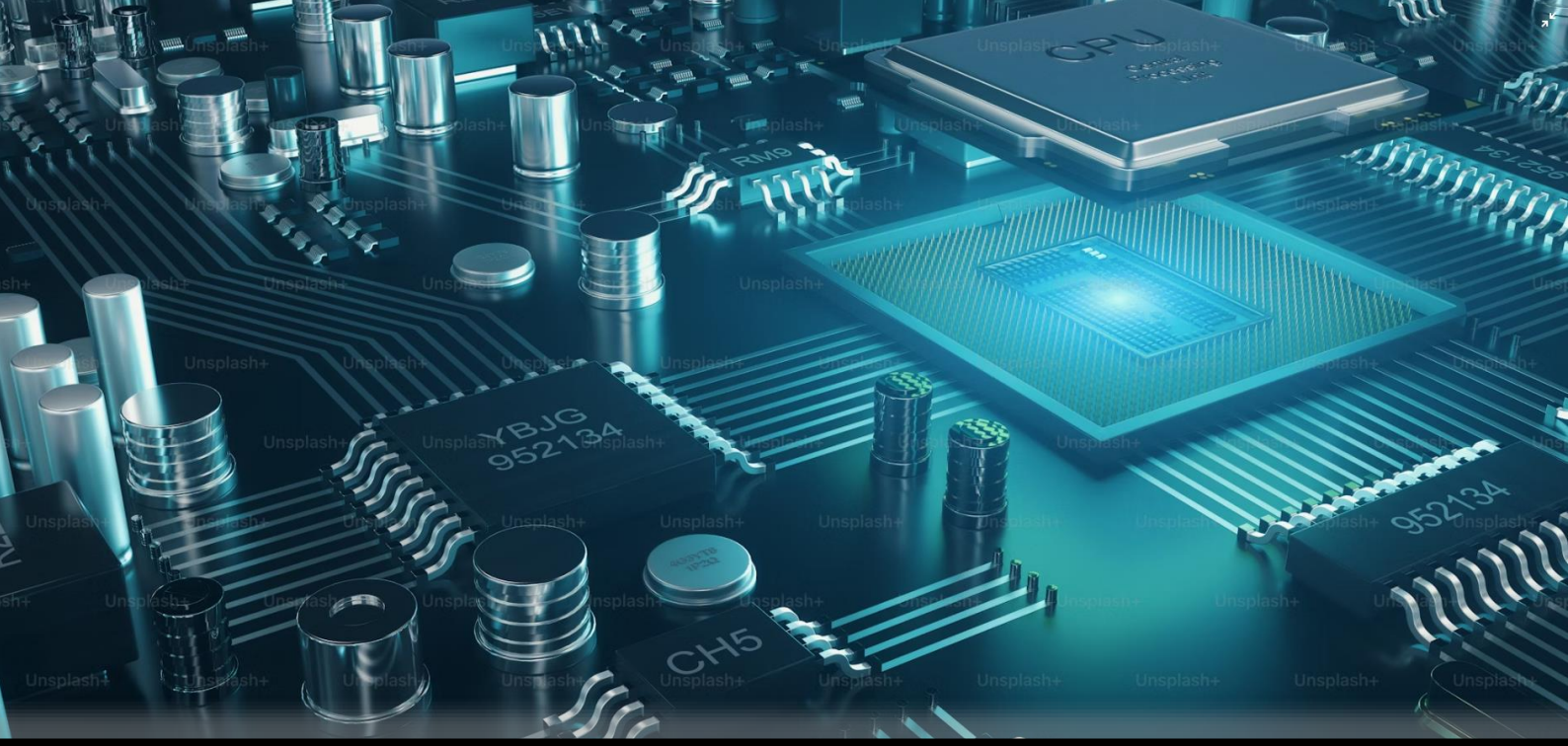
Python Code



```python
def choose_method_of_password(password):
    def by_keypad():
                print("Access granted: The password is correct...")
                print("done")
                open_servo()
                root.destroy()
                open_home_app(password)
                break
               elif response == "incorrect":
                print("Access denied: The password is incorrect, try again..")

    root = Tk()
    root.geometry("650x380")
    root.title("My Smart Home")
    logo = PhotoImage(file='SH2.png')
    root.iconphoto(True, logo)
    lbl = Label(root)
    player = tkvideo("background.mp4", lbl, loop=1)
    player.play()
    lbl.place(x=0, y=0)
    button1 =Button(root,
                    text="enter the password with Laptop",
                    width=30, height=3,font=("Arial",10,"bold"),bg="white", fg="black"
                    ,command=by_laptop)
    button2 =Button(root,
                    text="Enter the password with Keypad",
                    width=30, height=3,font=("Arial",10,"bold"),bg="white", fg="black"
                    ,command=by_keypad)
    button1.pack(pady=(100, 10))
    button2.pack(pady=(10, 40))
    root.mainloop()

# code start from here
with open("PASSWORD.txt", "r") as file:
    password = file.read().strip()


s = serial.Serial(port="COM5", baudrate=115200, timeout=0)
time.sleep(3)
if not s.is_open:
    s.open()
print("done")

main_window = Tk()
main_window.geometry("650x380")
main_window.title("House of Team 6")

logo = PhotoImage(file='SH2.png')
main_window.iconphoto(True, logo)
```

You can find all python script here in this link :

https://drive.google.com/file/d/1-fD6N1-p9-39994_AEn8AtkaRhF9lkFO/view?usp=drivesdk



Hardware Components

Design Item ID	Description
ATMEGA-8A	8KB 2.7V~5.5V AVR 16M
Buzzer	Buzzers ROHS
Capacitor	0402 Multilayer Ceramic
Crystal	16MHz 8pF ±10ppm ±5
Diode	Diode 1000 V 1A Surface
L293D-Driver	SOIC-20-300mil Stepper
LDR sensor	16kΩ~33kΩ 520nm 500
LM7805-Regu	83dB@(120Hz) 1A Fixed 5
Motor	DC Motor Vibration, ERM
NPN transistor	35@5mA,10V 202mW 1
NTC Thermisto	NTC Thermistor 100k 04
PIR SENSOR IN	Sensitive Element Interfac
Power connect	2 Position Non-Gendered
Push button	Round Button 50mA Stan
Resistor	62.5mW Thick Film Resist
RGB LED	0404 RGB LEDs ROHS
Slide switch	Slide Switch SPDT Surface
Temperature s	Temperature Sensor Digit
USB	USB Connectors ROHS
Yellow LED	20mA 104mcd Colorless

Footprints

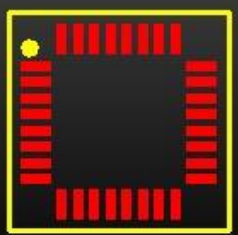
Name	Pads	Primiti...
ATMEGA-8A	32	214
BUZZER	4	30
CAPACITOR	2	14
CRYSTAL	4	15
DIODE	2	20

Place | Add | Delete | Edit

Footprint Primitives

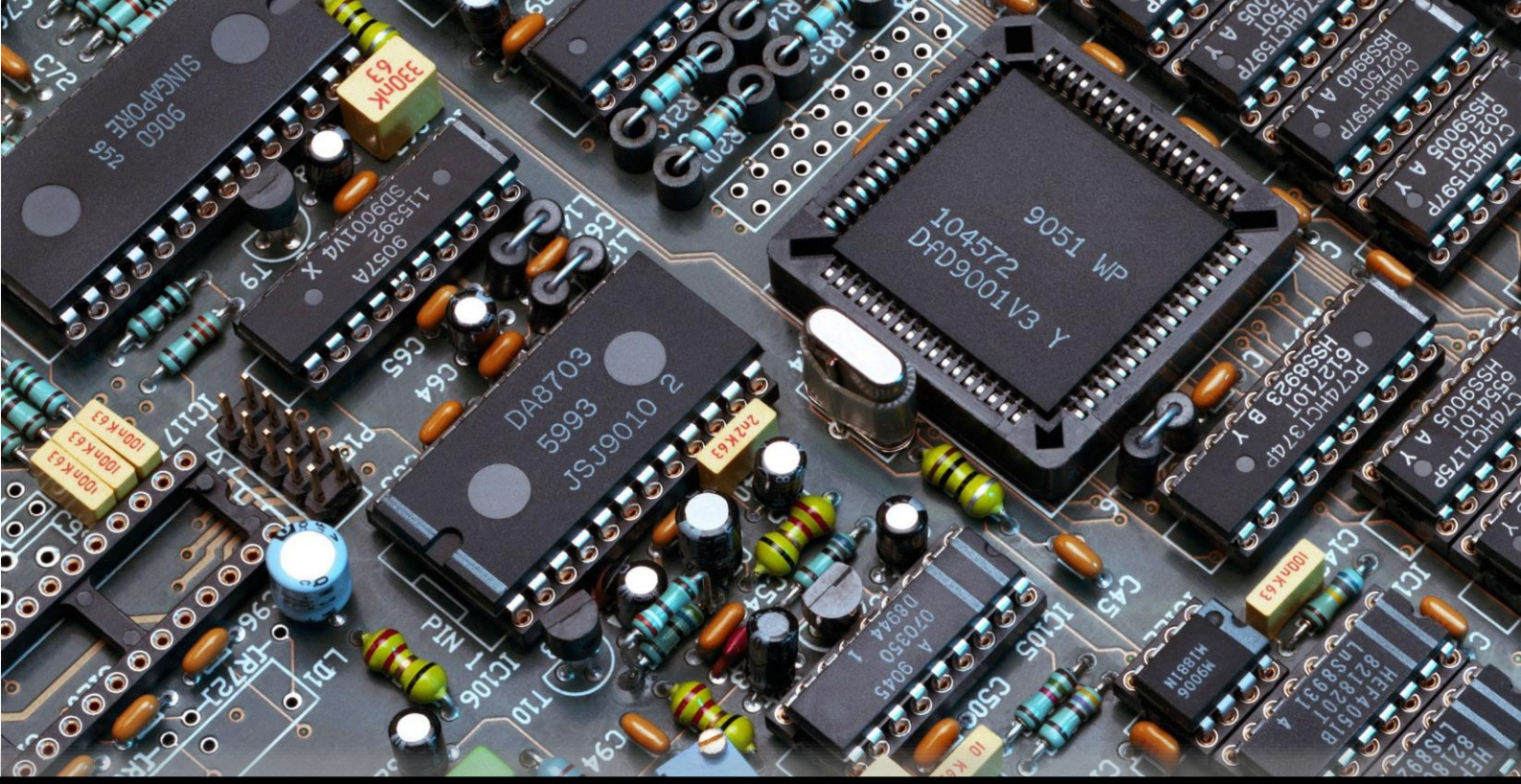
Type	N...	X-Size	Y-Size	Layer
Arc		0.25...		Top...
Track		0mm		Mec...
Track		0mm		Mec...
Track		0mm		Mec...
Track		0mm		Mec...
Track		0mm		Mec...

Other

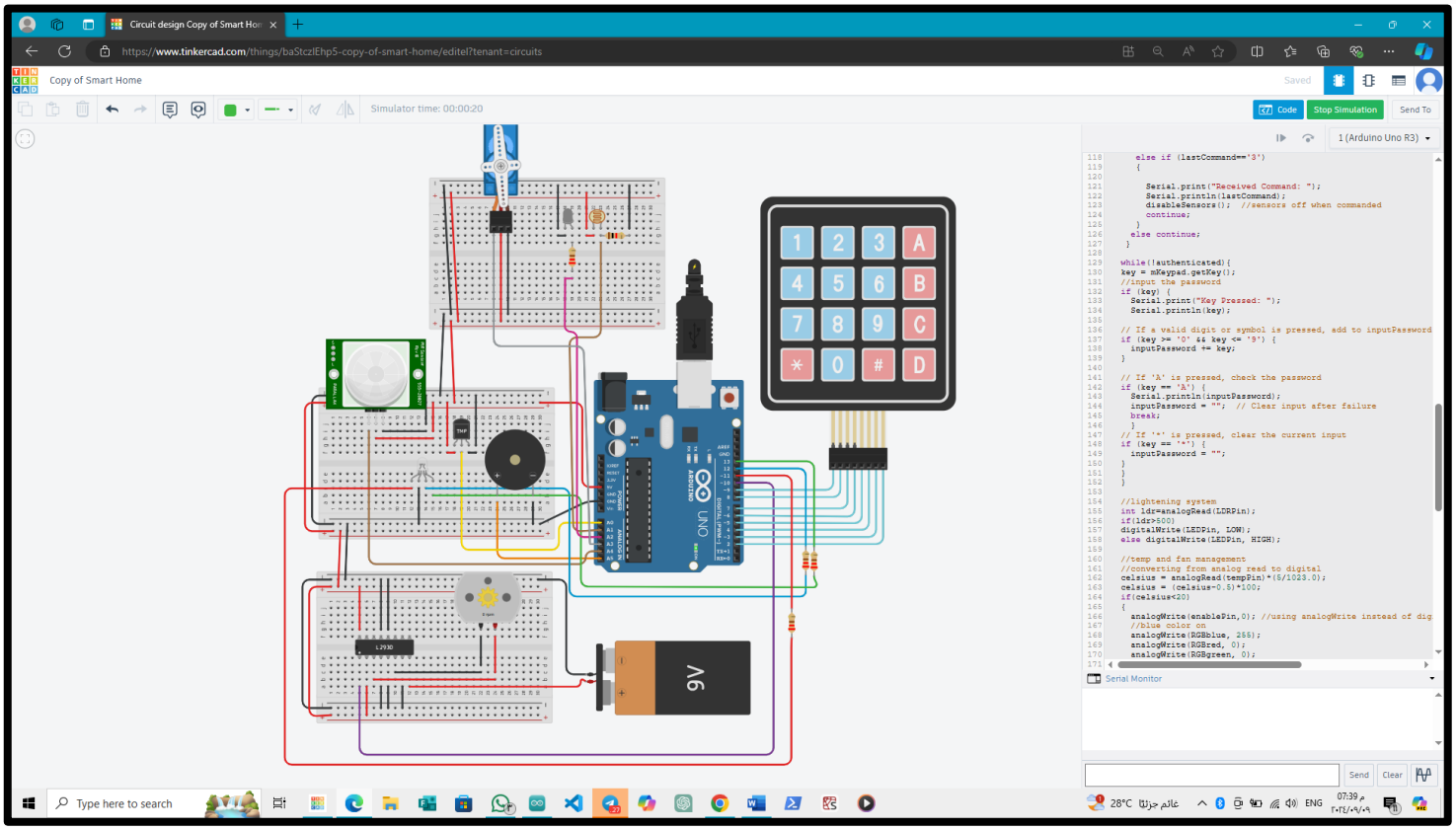


Part List

	Component	Mfr. Part #
01	Resistor	RC0402JR-0710KL
02	Ceramic Disk Capacitor	CL05B104KP5NNNC
03	1N4007 Diode	S1MFL
04	LED	LTST-C190KSKT
05	ATmega8a-AN 8 bit MCU	ATMEGA8A-AN
06	7805 Voltage Regulator	NCV7805BDTRKG
07	Power Connector 2 Way	ASMPR45-1X2-RK
08	Buzzer (relay)	CMT-8504-100-SMT-TR
09	On/Off Switch	CL-SB-12B-01T
10	NPN Transistor	MUN5211T1G
11	Push buttons	PTS647SN50SMTR2LFS
12	NTC Thermistor	NCP15WL104J03RC
13	Pir sensor	LS6512-S
14	USB	UJ2-AV-1-SMT-TR
15	LDR sensor	PDV-P8103
16	Temperature sensor	LM75AIM/NOPB
17	Fan blades	N/A
18	Motor	HD-EMB1305-SC
19	Servo motor	N/A
20	Polarized plug	N/A
19	Crystal	ABM8W-16.0000MHZ-8-K1Z-T3
20	L293D-Driver	L293DD013TR
21	RGB LED	UHD1110-FKA CL1A13R3Q1BBQFMF3



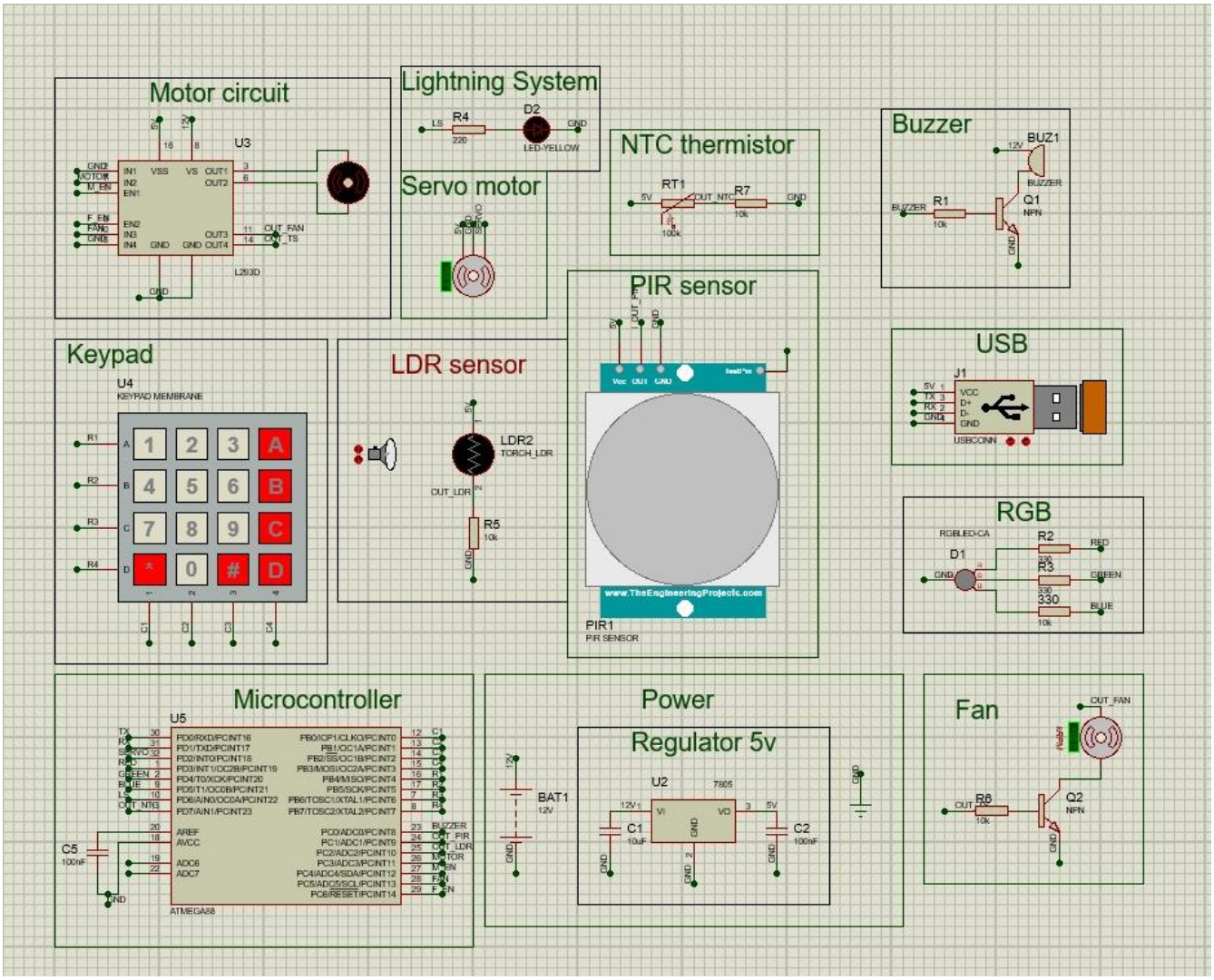
Tinkercad Simulation

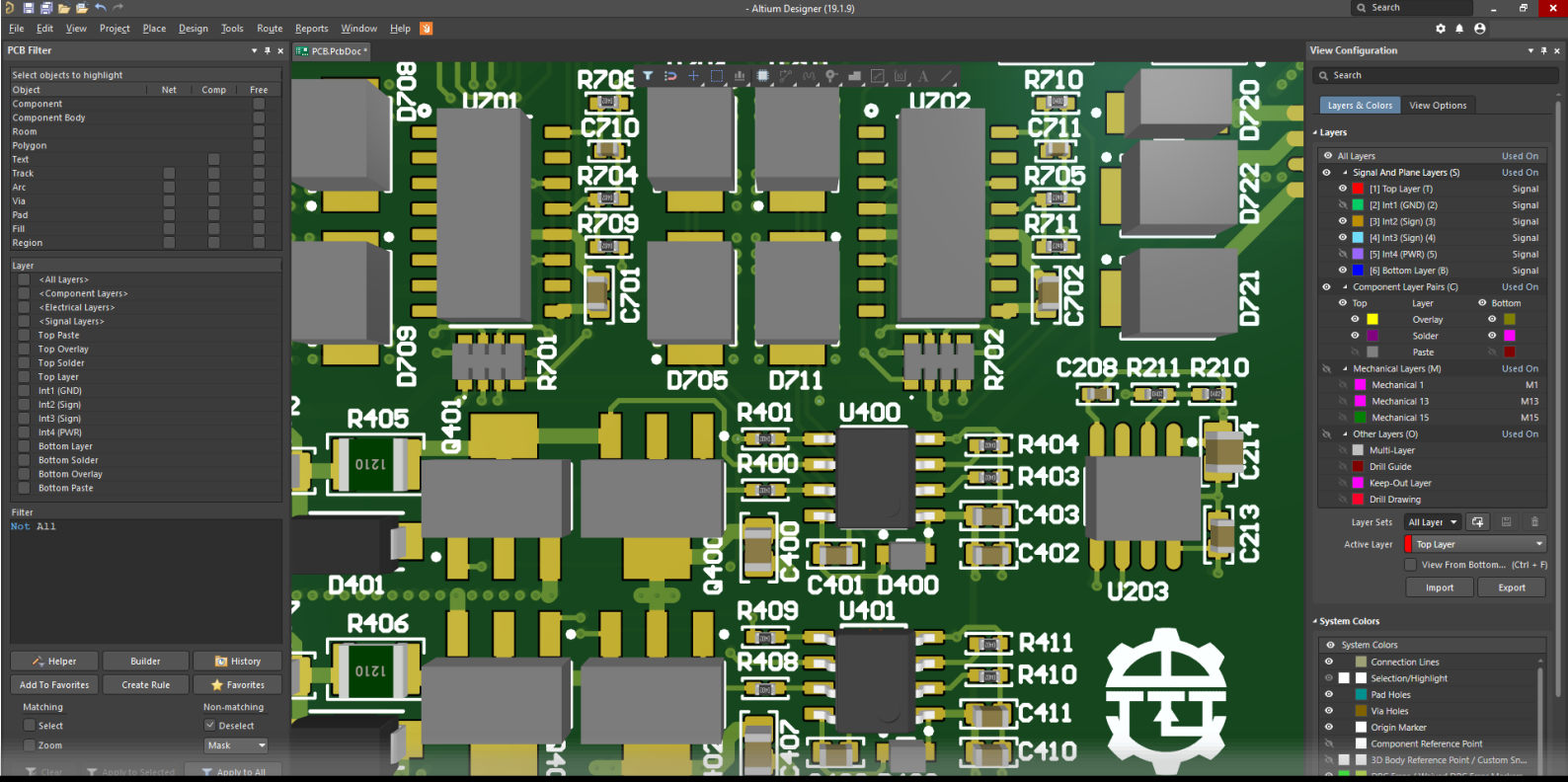


You can find link of simulation here : [Smart Home (tinkercad.com)]

Proteus

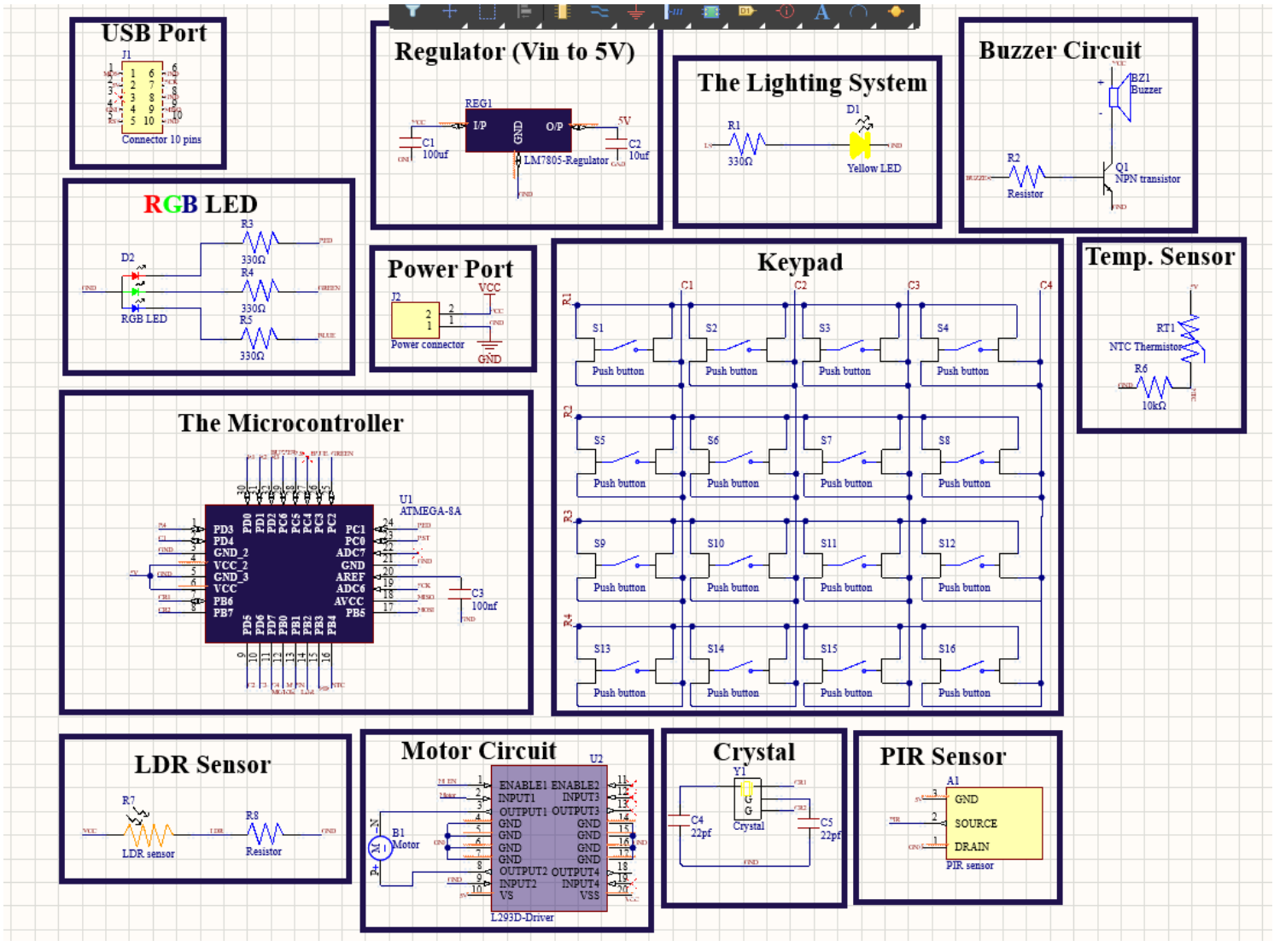
First part : Schematic



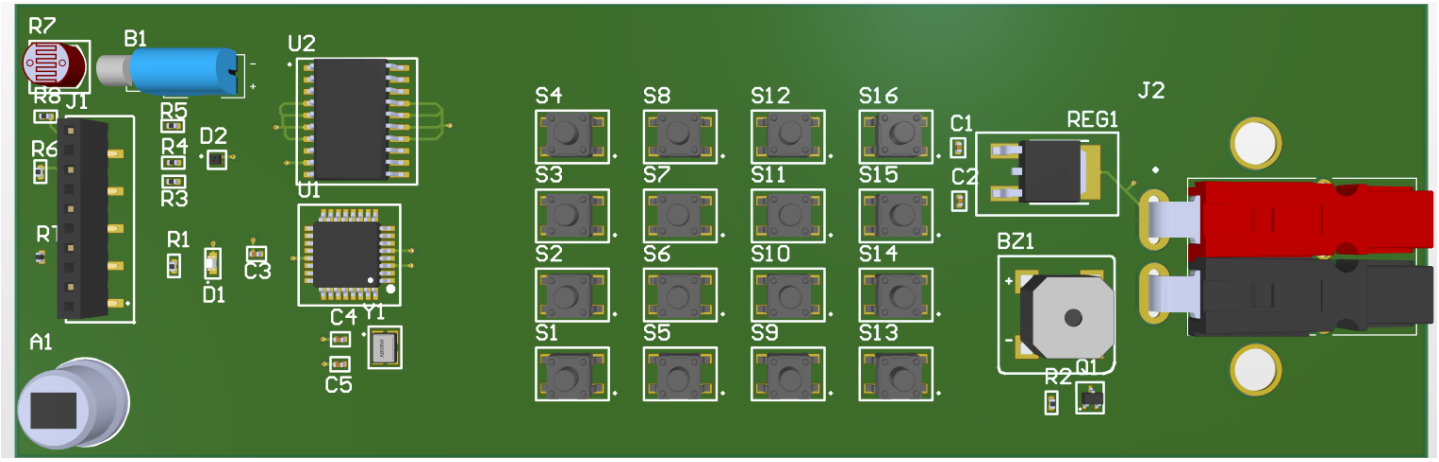


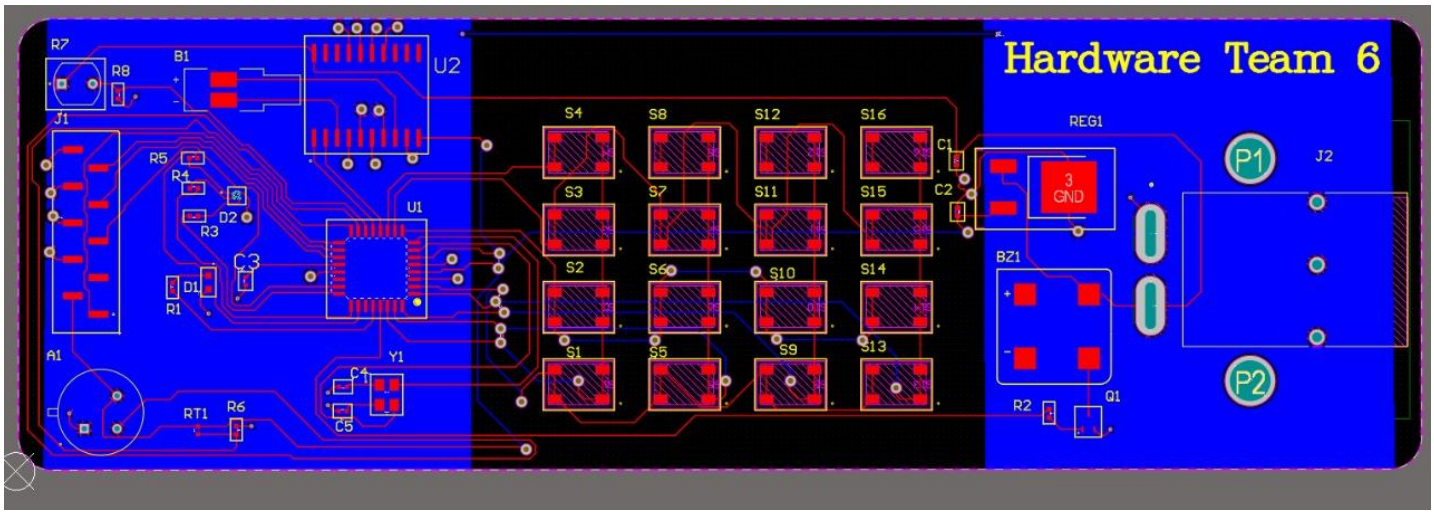
Altium Design

first part : Schematic Design

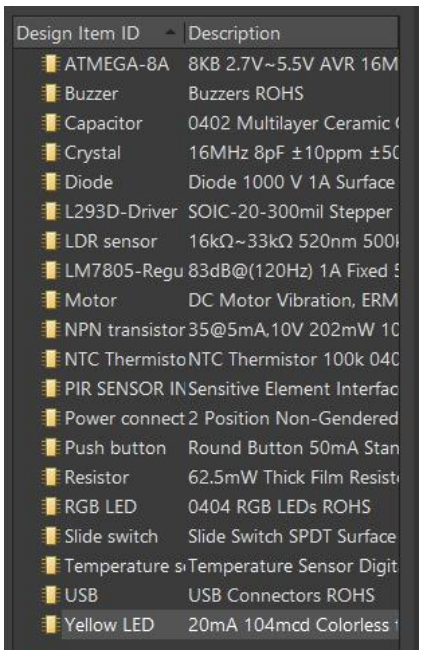


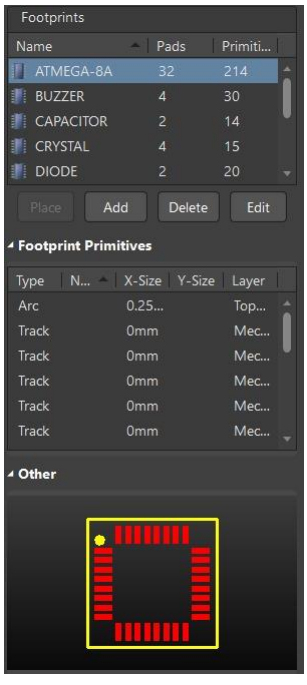
Second part : PCB Design

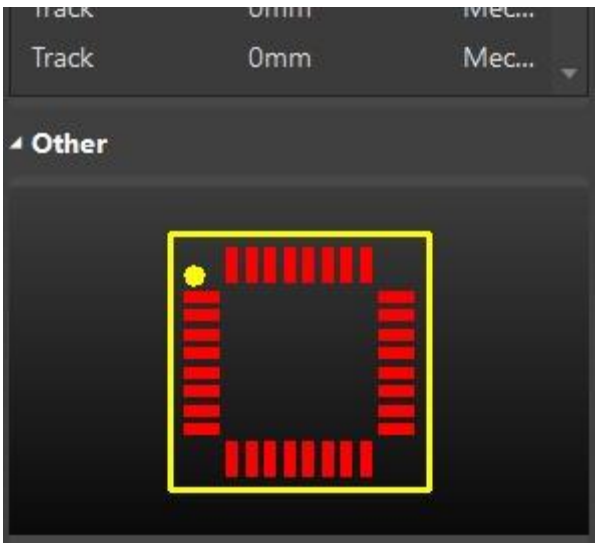




Third part : Components and libraries

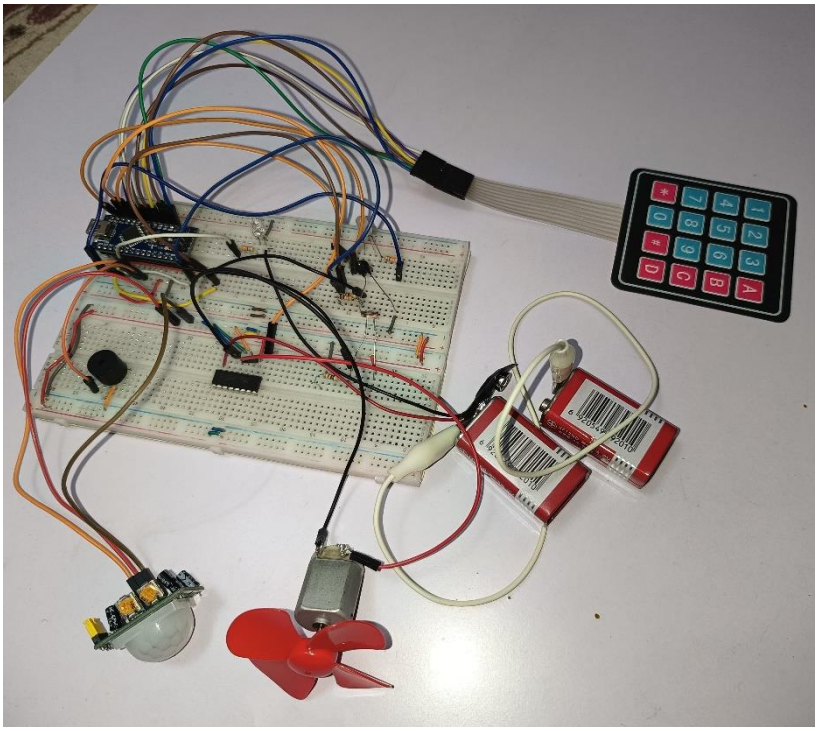






Fourth part : Fabricated circuit

with Arduino :



With ATmega :