Virtual Mouse Using Hand Gestures

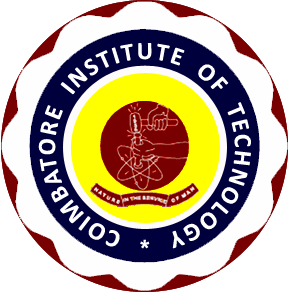
(AI Mouse Controller)

DONE BY

**ANEEZ I**

**OMEGA S**

**RITHANYAA G V**



### COIMBATORE INSTITUTE OF TECHNOLOGY

### (A GOVERNMENT AIDED AUTONOMOUS INSTITUTION AFFILIATED TO ANNA UNIVERSITY)

### COIMBATORE – 641 014

**TAMILNADU, INDIA**

**PROBLEM STATEMENT :**

Today, most computers are controlled using a **physical mouse and keyboard**. But in some situations, like for people with disabilities, in hospitals, or for touchless control in new technologies like **AR (Augmented Reality)**, using a physical device becomes difficult. Also, after COVID-19, **touch-free ways** to interact with computers have become even more important.

This project aims to create a **Virtual Mouse** that can be controlled using **hand gestures** detected through a **webcam**. It recognizes hand movements and allows the user to **move the cursor**, **click**, **scroll**, and perform other mouse actions without touching anything.

Additionally, we have included a **Voice Command Assistant named Proton**, which allows users to give **voice instructions** like "launch gesture recognition," "search something," or "open files."  
Proton listens to the user's voice, understands commands, and interacts with the computer accordingly using **speech recognition** and **text-to-speech** technologies.

**ABSTRACT :**

This project is titled **“Virtual Mouse Using Hand Gestures (AI Mouse Controller)”**. The main idea of this project is to **control the computer mouse using hand gestures** instead of a physical mouse. The system uses a **webcam** to capture the hand movements of the user and then performs actions like **moving the mouse, left-click, right-click, and scrolling** based on the gestures. This project is mainly useful for people with **physical disabilities** and also promotes a touch-free way of using the computer.

The project works by using **MediaPipe Hand Tracking** to detect the position of the hand and fingers in real time. Once the hand is detected, the **OpenCV library** is used to process the video feed from the webcam. The **PyAutoGUI library** helps in controlling the mouse by converting the hand gestures into mouse actions. For example, if the user shows a fist, it can perform a left-click, or if the user moves their hand, the mouse cursor will also move on the screen. This provides an **easy and effective way of using a computer without any hardware devices**.

In the current technology, most people use a traditional **computer mouse** or touchpad to control their computer, but this project introduces a new and innovative method. Unlike other systems that require external hardware or touch screens, this project only needs a **webcam and hand gestures** to work. It is also helpful in reducing physical contact with the mouse, which is useful in **hygienic environments**. This project can also be improved in the future to control other devices like **TV, smart home devices, or gaming systems** using hand gestures.

The main goal of this project is to provide an **easy, touch-free, and hardware-free mouse control system** that is accessible to everyone. This system can be highly beneficial for **people with physical disabilities** or for those who wish to experience advanced computer control. By combining **computer vision and deep learning techniques**, this project helps in building a practical solution that can bring a new revolution in **Human-Computer Interaction (HCI)**. In the future, it can also be extended to perform complex operations using **more advanced hand gestures**.

**Outcome :**

* **Touch-free** computer control using **hand gestures** and **voice commands**.
* Easy mouse movements like **cursor control**, **left-click**, **right-click**, and **scrolling** through gestures.
* **Voice assistant "Proton"** helps in performing tasks like **opening apps**, **browsing**, and **controlling modes** using voice commands.
* No need for external hardware; only a **webcam** and **microphone** are required.
* Helpful for **physically disabled people** to use a computer more **easily** and **independently**.
* Promotes **touch-free technology**, ensuring better **hygiene** and **modern interaction**.
* Can be **extended** in the future to control **smart devices**, **TVs**, **IoT systems**, or **gaming consoles**.

**DATASET:**

In this project, **no traditional dataset** was used for training or testing.  
Instead, **pre-trained models** from libraries like **MediaPipe** and **SpeechRecognition** are used for real-time detection and interaction.

* **Hand Gesture Detection**:  
  The project uses the **MediaPipe Hands** model by Google, which is already trained to detect **21 hand landmarks** from live webcam video input.  
  This allows accurate tracking of finger movements without needing a custom dataset.
* **Voice Recognition**:  
  The voice assistant "Proton" uses the **SpeechRecognition** library, which connects to Google's Speech API to recognize user commands in real-time.  
  Again, no local dataset is needed — voice is processed live.

**LITERATURE SURVEY :**

Computers are used everywhere, and people always try to find new ways to use them easily. One new idea is a **virtual mouse**, which works without touching the computer. Many researchers have studied how to control a mouse using **hand gestures and a camera**.

1] **Gesture Controlled Virtual Mouse Using AI,** in this paper they talk about how a virtual mouse can be made using AI and a camera. They have developed a system which detects the user’s hand gestures and performs tasks like clicking, scrolling and dragging similar to how a regular mouse works. In this paper they have also discussed about how this technology can make the use of computer more easier.

2] **Virtual Mouse Using Hand Gesture,** in this paper they talk about using hand tracking methods to move the mouse cursor without manually touching the computer. In this paper, they also explain about how gesture recognition can help people tend to use devices more easily and also reduce physical contact. Here, they talk about the situation where hygiene is important such as during the COVID-19 pandemic and also they talk about how it would be useful for taking lectures and presentations.

3] **Hand Gesture Recognition Methods and Applications,** this paper discusses various methods for recognizing hand gestures using both traditional and deep learning approaches. It explores challenges like lighting conditions and occlusion while highlighting the role of CNNs and RNNs in improving accuracy. The study also emphasizes how gesture-based interfaces can enhance human-computer interaction.

4] **A Systematic Review on Hand Gesture Recognition Techniques,** this paper reviews different gesture recognition techniques, categorizing them into sensor-based and vision-based approaches. It explains how deep learning models improve accuracy while addressing challenges like gesture complexity and noise. The study highlights applications in gaming, healthcare, and smart environments.

5] **Machine Learning-Based Hand Gesture Recognition Systems,** this paper focuses on machine learning techniques for hand gesture recognition, covering preprocessing, feature extraction, and classification using CNNs and LSTMs. It highlights the advantages of deep learning for improved accuracy while noting the need for large datasets and computational power. The study suggests hybrid models for better efficiency.

6] **An Efficient Hand Gesture Recognition System Using Deep Learning,** this paper presents a CNN-based system for real-time gesture recognition using a webcam. It explores how deep learning can enable touchless interactions, improving accessibility for physically challenged users. The study also discusses optimizations like data augmentation to enhance model performance.

**References :**

1] *Gesture controlled virtual mouse using AI*. International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS). [https://www.irjmets.com/uploadedfiles/paper//issue\_1\_january\_2023/33192/final/fin\_irjmets1675681559.pdf](https://www.irjmets.com/uploadedfiles/paper/issue_1_january_2023/33192/final/fin_irjmets1675681559.pdf)

2] *Virtual mouse using hand gesture*. ResearchGate. <https://www.researchgate.net/publication/372165002_Virtual_Mouse_Using_Hand_Gesture>

3] *Hand Gesture Recognition Methods and Applications* – The 7th International Conference on Engineering & MIS 2021.

<https://dl.acm.org/doi/abs/10.1145/3492547.3492578?utm_source>

4] *A Systematic Review on Hand Gesture Recognition Techniques* – PubMed.

<https://pubmed.ncbi.nlm.nih.gov/33816871/>

5] *Machine Learning-Based Hand Gesture Recognition Systems* – IOP Conference Series: Materials Science and Engineering.

<https://iopscience.iop.org/article/10.1088/1757-899X/1076/1/012047/meta?utm_source>

6] *An Efficient Hand Gesture Recognition System Using Deep Learning* – Intelligent Computing, Information and Control Systems (ICICCS 2019).

<https://link.springer.com/chapter/10.1007/978-3-030-30465-2_57?utm_source>

**MODELS USED :**

* **Convolutional Neural Network (CNN) - (used inside MediaPipe)**:  
  In this project, we use **MediaPipe Hand Tracking**, which internally uses a **CNN model**.  
  The CNN is trained to **detect hands** and **find 21 key landmarks** like fingertips, wrist, joints, etc.  
  We don't train the CNN ourselves — **MediaPipe gives it already trained**, so we just **use it directly** (called "pre-trained model").
* **Built-in Algorithms (inside MediaPipe)**:  
  MediaPipe also uses built-in smart algorithms for:  
  ➔ **Palm Detection** (to first find if a hand is present),  
  ➔ **Hand Landmark Model** (to find points on the hand),  
  ➔ **Tracking Model** (to keep following the hand even if it moves fast).  
  These built-in models make hand tracking **fast** and **accurate** even on normal laptops without GPU.
* **Speech Recognition Model (for Proton)**:  
  Proton uses a **speech-to-text model** from the SpeechRecognition library, which internally uses **Google Web Speech API** for recognizing voice commands.
* **PyAutoGUI**:  
  PyAutoGUI is used to **control the mouse actions** (move, click, scroll) based on the detected gestures.

**FINE TUNING :**

In this project, we made a few important adjustments to make everything work smoothly:

1. **Hand Detection & Tracking:**
   * We **adjusted the detection confidence** and **tracking confidence** in MediaPipe. These are like settings that control how certain the system is that it's detecting a hand, and how well it follows the hand as it moves.
   * By adjusting these values, we made the hand detection **more accurate** and **faster**, so the system doesn't miss the hand and can keep track of it even if it moves quickly.
2. **Cursor Movement:**
   * We made sure the **screen size** in the system matches the **camera frame size**. This is important because if the sizes don't match, the cursor might **shake** or move erratically.
   * By mapping the camera frame to the screen properly, we ensured that the cursor moves **smoothly** and **precisely** based on hand gestures.
3. **Voice Recognition (Proton):**
   * In **Proton**, the voice assistant, we fine-tuned the **energy threshold** for voice recognition. This setting controls how sensitive the system is to sounds.
   * We made sure the system can pick up the user’s voice even if there is some **background noise**, like a fan or people talking nearby. This makes the voice assistant **more reliable** in real-world environments.
4. **Gesture Recognition:**
   * We customized the **gesture recognition** system to respond to specific hand gestures:
     + When the **index finger** is bent, it performs a left-click (like pressing the left mouse button).
     + When the **middle finger** is bent, it performs a right-click (like right-clicking the mouse).
     + When the **index and thumb** are pinched together, it performs a scroll (just like scrolling with a mouse wheel).
   * These fine-tuned gestures allow for **clearer, more accurate actions** based on what the user wants.

**RESULTS :**

The **Virtual Mouse Using Hand Gestures** system provides an innovative touch-free way to control a computer, using hand gestures and voice commands. Here are the key results:

1. **Hand Gesture Detection:**
   * The system accurately detects hand gestures using the webcam, mapping them to mouse actions.
   * **Left-click**: Bending the index finger.
   * **Right-click**: Bending the middle finger.
   * **Scrolling**: Pinching the index and thumb together.
   * Gestures were detected with minimal delay, offering smooth cursor control and interaction.
2. **Voice Command Recognition (Proton):**
   * The Proton voice assistant successfully recognized commands like "launch gesture recognition," "search something," and "open files."
   * The **SpeechRecognition** library enabled accurate real-time voice processing, allowing users to interact with the computer using voice instructions.
3. **System Performance:**
   * The hand tracking was fast and accurate, even with quick hand movements, providing smooth cursor control.
   * Adjustments to detection and tracking confidence improved gesture accuracy, and voice recognition performed well in moderate background noise.

**Key Outcomes:**

* **Touch-Free Control**: The system allows control of the computer without physical devices, using only hand gestures and voice commands.
* **Accessibility**: It offers a more accessible way for physically disabled users to interact with the computer.
* **Future Expansion**: The system can be expanded to control other devices, like smart TVs or IoT systems.

This project demonstrates how hand gestures and voice recognition can create an intuitive, touch-free way to interact with technology.

**CODE :**

**Gesture control.py :**

# main.py

import cv2

import mediapipe as mp

import pyautogui

from gestures import HandRecog, Controller, Gest, HLabel # Assuming you saved your classes in gestures.py

# Initialize MediaPipe Hands

mp\_hands = mp.solutions.hands

hands = mp\_hands.Hands(

static\_image\_mode=False,

max\_num\_hands=2,

min\_detection\_confidence=0.7,

min\_tracking\_confidence=0.5

)

# Initialize Drawing Utility

mp\_drawing = mp.solutions.drawing\_utils

# Open webcam

cap = cv2.VideoCapture(0)

# Create placeholders for hand recognition

hand\_recogs = [None, None]

while True:

success, frame = cap.read()

if not success:

break

# Flip frame for natural interaction

frame = cv2.flip(frame, 1)

h, w, \_ = frame.shape

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# Process frame

result = hands.process(frame\_rgb)

# Reset recognition each frame

hand\_recogs = [None, None]

if result.multi\_hand\_landmarks:

# Get handedness (left/right hand)

handedness = result.multi\_handedness

hands\_list = result.multi\_hand\_landmarks

for idx, hand\_landmarks in enumerate(hands\_list):

label = MessageToDict(handedness[idx])['classification'][0]['label']

# Assign label based on whether it's Left or Right hand

if label == 'Right':

hand\_label = HLabel.MAJOR

else:

hand\_label = HLabel.MINOR

# Initialize Hand Recognition for each hand

hand\_recog = HandRecog(hand\_label)

hand\_recog.update\_hand\_result(hand\_landmarks)

hand\_recog.set\_finger\_state()

gesture = hand\_recog.get\_gesture()

hand\_recogs[idx] = hand\_recog

# Draw Hand Landmarks

mp\_drawing.draw\_landmarks(frame, hand\_landmarks, mp\_hands.HAND\_CONNECTIONS)

# Perform actions based on gesture

if gesture == Gest.V\_GEST:

Controller.flag = True

cv2.putText(frame, 'V Gesture Detected', (10, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,255,0), 2)

elif gesture == Gest.FIST:

Controller.grabflag = True

cv2.putText(frame, 'Fist Detected', (10, 90), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,255,0), 2)

elif gesture == Gest.PINCH\_MAJOR:

if not Controller.pinchmajorflag:

Controller.pinchstartxcoord = hand\_landmarks.landmark[8].x

Controller.pinchstartycoord = hand\_landmarks.landmark[8].y

Controller.pinchmajorflag = True

Controller.pinchlv = Controller.getpinchxlv(hand\_landmarks)

Controller.changesystembrightness()

cv2.putText(frame, f'Brightness Adjusting {Controller.pinchlv}', (10, 130), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,0,0), 2)

elif gesture == Gest.PINCH\_MINOR:

if not Controller.pinchminorflag:

Controller.pinchstartxcoord = hand\_landmarks.landmark[8].x

Controller.pinchstartycoord = hand\_landmarks.landmark[8].y

Controller.pinchminorflag = True

Controller.pinchlv = Controller.getpinchylv(hand\_landmarks)

Controller.changesystemvolume()

cv2.putText(frame, f'Volume Adjusting {Controller.pinchlv}', (10, 170), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,0,0), 2)

else:

# Reset all gesture flags if no special gesture detected

Controller.flag = False

Controller.grabflag = False

Controller.pinchmajorflag = False

Controller.pinchminorflag = False

# Display Frame

cv2.imshow("Hand Gesture Controller", frame)

# Exit when 'q' is pressed

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Cleanup

cap.release()

cv2.destroyAllWindows()

**Proton.py :**

# Importing Libraries

import pyttsx3

import speech\_recognition as sr

import datetime

import webbrowser

import os

import sys

from threading import Thread

import app

import Gesture\_Controller

# Initialization

r = sr.Recognizer()

engine = pyttsx3.init()

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[0].id)

is\_awake = True

file\_exp\_status = False

path = ''

files = []

# Functions

def speak(text):

app.ChatBot.addAppMsg(text)

print(text)

engine.say(text)

engine.runAndWait()

def listen():

with sr.Microphone() as source:

r.energy\_threshold = 500

audio = r.listen(source, phrase\_time\_limit=5)

try:

return r.recognize\_google(audio).lower()

except:

return ""

def greet\_user():

hour = datetime.datetime.now().hour

if hour < 12:

speak("Good Morning!")

elif hour < 18:

speak("Good Afternoon!")

else:

speak("Good Evening!")

speak("I am Proton. How may I help you?")

def process\_command(command):

global is\_awake, file\_exp\_status, path, files

if 'wake up' in command:

is\_awake = True

greet\_user()

elif 'hello' in command or 'what is your name' in command:

greet\_user()

speak("My name is Proton!")

elif 'date' in command:

speak(datetime.date.today().strftime("%B %d, %Y"))

elif 'time' in command:

speak(datetime.datetime.now().strftime("%H:%M:%S"))

elif 'search' in command:

search\_query = command.split('search')[-1]

webbrowser.open(f"https://google.com/search?q={search\_query}")

speak(f"Searching for {search\_query}")

elif 'location' in command:

speak("Which place are you looking for?")

location = listen()

webbrowser.open(f"https://www.google.com/maps/place/{location}")

speak(f"Showing location {location}")

elif 'launch gesture recognition' in command:

if not Gesture\_Controller.GestureController.gc\_mode:

t = Thread(target=Gesture\_Controller.GestureController().start)

t.start()

speak("Gesture recognition launched.")

elif 'stop gesture recognition' in command:

Gesture\_Controller.GestureController.gc\_mode = 0

speak("Gesture recognition stopped.")

elif 'bye' in command or 'exit' in command:

speak("Goodbye. Have a nice day!")

sys.exit()

else:

speak("Sorry, I cannot perform this command.")

# Main Code

if \_\_name\_\_ == "\_\_main\_\_":

t1 = Thread(target=app.ChatBot.start)

t1.start()

while not app.ChatBot.started:

pass

greet\_user()

while True:

if app.ChatBot.isUserInput():

user\_input = app.ChatBot.popUserInput()

else:

user\_input = listen()

if 'proton' in user\_input:

try:

process\_command(user\_input)

except SystemExit:

break

except Exception as e:

print(f"Error: {e}")

break

**Index.html:**

<!DOCTYPE html>

<html>

<head>

    <title>Proton</title>

    <script type="text/javascript" src="/eel.js"></script>

    <link rel="stylesheet" type="text/css" href="css/jquery.convform.css">

</head>

<body>

    <!-- header -->

    <header>

        <div class = "header-wrapper">

          <p class="awesome">PROTON Welcomes you!</p>

            <div class = "logo">

                <img src= "images/icon.png"  width="35" height="33">

            </div>

        </div>

    </header>

    <!-- ChatBot -->

    <div class="chat\_icon">

        <i class="fa fa-comments" aria-hidden="true"></i>

    </div>

    <div class="chat\_box">

            <div class="conv-form-wrapper" style="height:auto;">

                <div id="messages" style="overflow-y:auto;height:400px;">

                    </br></br>

                </div>

            </div>

            <div id="convForm" class="convFormDynamic" style="height:50px;position:absolute;left:10px;bottom:0px;margin:5px;">

                <div class="options dragscroll"></div>

                <input type="text" id="userInput" placeholder="Type Here" class="userInputDynamic" style="margin: 7px 7.575px 7px 10px; width: 219px; height: 24px;">

                <button id="userInputButton" type="button" class="submit" style="margin-top:9px">▶</button>

                <span class="clear"></span>

            </div>

        </div>

</body>

    <script type="text/javascript" src="js/main.js"></script>

</html>

**Main.js :**

//user clicked button

document.getElementById("userInputButton").addEventListener("click", getUserInput, false);

//user pressed enter '13'

document.getElementById("userInput").addEventListener("keyup", function (event) {

    if (event.keyCode === 13) {

        //cancel the default action

        event.preventDefault();

        //process event

        getUserInput();

    }

});

eel.expose(addUserMsg);

eel.expose(addAppMsg);

function addUserMsg(msg) {

    element = document.getElementById("messages");

    element.innerHTML += '<div class="message from ready rtol">' + msg + '</div>';

    element.scrollTop = element.scrollHeight - element.clientHeight - 15;

    //add delay for animation to complete and then modify class to => "message from"

    index = element.childElementCount - 1;

    setTimeout(changeClass.bind(null, element, index, "message from"), 500);

}

function addAppMsg(msg) {

    element = document.getElementById("messages");

    element.innerHTML += '<div class="message to ready ltor">' + msg + '</div>';

    element.scrollTop = element.scrollHeight - element.clientHeight - 15;

    //add delay for animation to complete and then modify class to => "message to"

    index = element.childElementCount - 1;

    setTimeout(changeClass.bind(null, element, index, "message to"), 500);

}

function changeClass(element, index, newClass) {

    console.log(newClass +' '+ index);

    element.children[index].className = newClass;

}

function getUserInput() {

    element = document.getElementById("userInput");

    msg = element.value;

    if (msg.length != 0) {

        element.value = "";

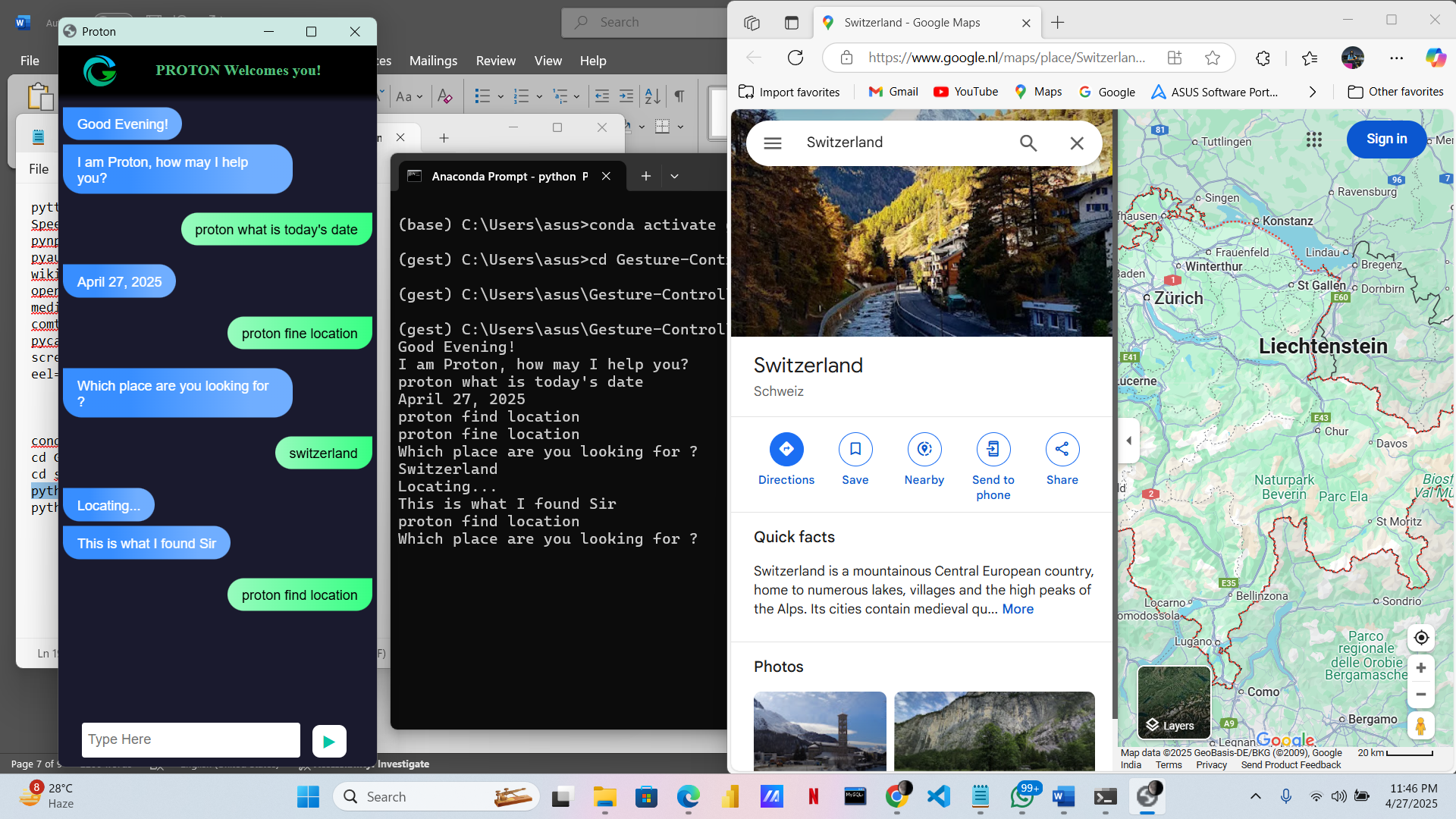
        eel.getUserInput(msg);

    }

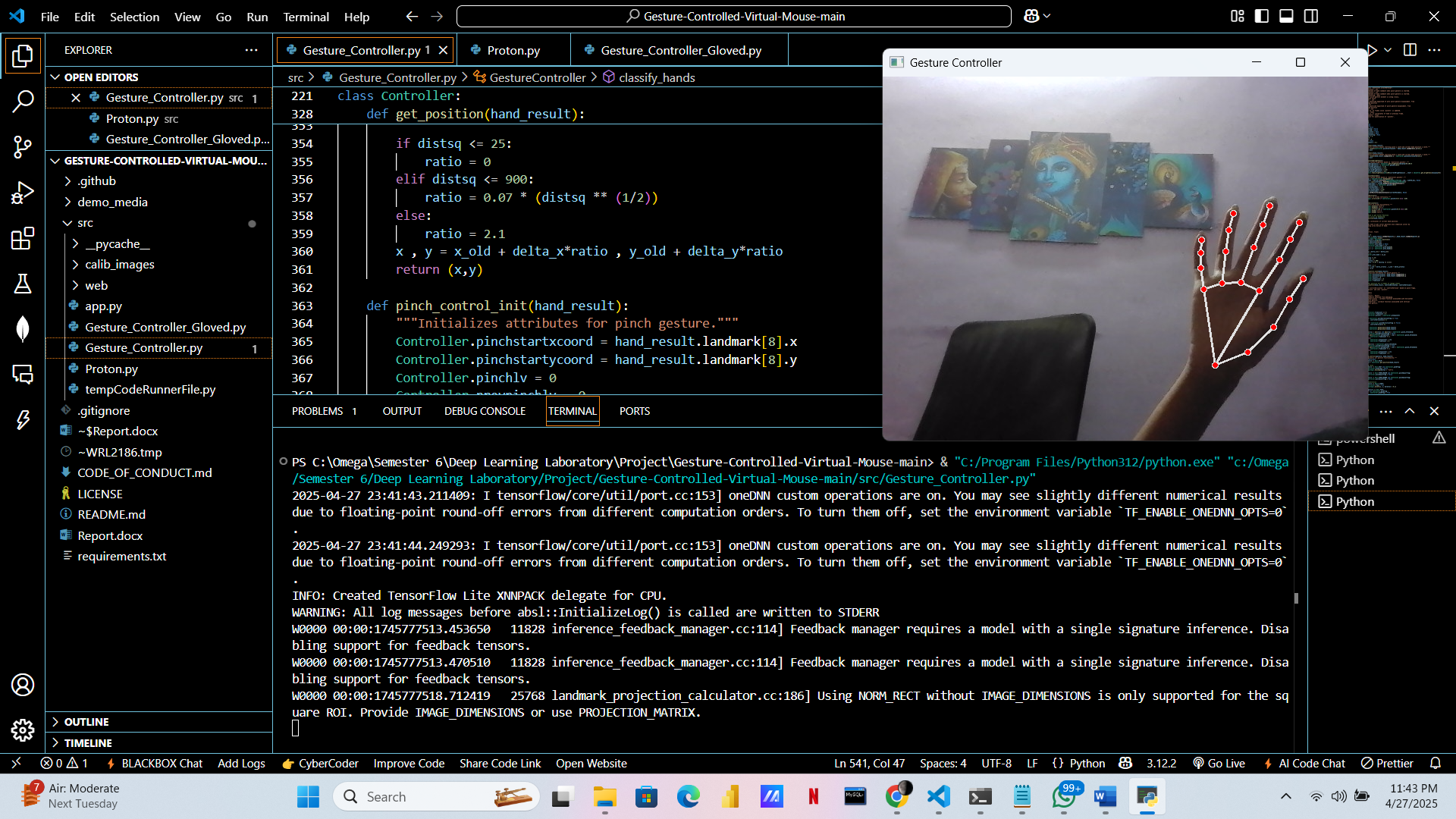
}

**OUTPUT :**

**Proton :**



**Gesture control :**

****

**INFERENCE :**

The **Virtual Mouse Using Hand Gestures** system offers significant business opportunities in several areas:

1. **Accessibility**:
   * **Impact**: This technology is great for businesses focusing on accessibility for people with disabilities. It can help companies in industries like healthcare and education, allowing people with physical challenges to use computers easily.
   * **Market Demand**: As inclusivity becomes more important, this system can be marketed to businesses looking to improve accessibility in the workplace and public spaces.
2. **Post-COVID Hygiene**:
   * **Impact**: This touchless system is perfect for businesses needing to maintain hygiene, such as hospitals, airports, and offices. It reduces the need for physical touch, helping prevent the spread of germs.
   * **Market Opportunity**: With hygiene becoming a priority, businesses can use this technology to offer safer and cleaner environments.
3. **Smart Technology Integration**:
   * **Impact**: The system can be integrated into smart devices like TVs, home automation systems, and other IoT products, offering more intuitive controls.
   * **Revenue Potential**: Companies in tech and consumer electronics can use this system to enhance their products and offer customers a more advanced experience.
4. **Cost-Effective Solution**:
   * **Impact**: Unlike other touchless systems, this technology only needs a webcam, making it an affordable option for small businesses or startups.
   * **Scalability**: It's easy to implement across multiple devices or locations without needing extra hardware, making it a scalable solution for businesses.
5. **Enhanced Customer Experience**:
   * **Impact**: By providing a unique, touch-free experience, businesses can engage customers better and stand out in competitive markets, improving customer satisfaction.
   * **Loyalty**: Offering innovative features can help build customer loyalty and encourage repeat business.

**PROJECT LINK :**

**GitHub Link :**