Hand Gesture Recognition using Arduino and OpenCV

B. TECH in COMPUTER SCIENCE & ENGINEERING

UE21CS251B – MPCA

SECTION-D

Ganesh Mayya K	PES1UG21CS198
Gautham P Atreyas	PES1UG21CS206
Gagan H R	PES1UG21CS196
Gudekote Karan Goud	PES1UG21CS211

1. PROBLEM STATEMENT

Hand gesture recognition is a challenging task with various practical applications such as sign language translation, human-computer interaction, and robotics. However, there are limited affordable and accessible systems available for real-time hand gesture recognition.

This project aims to develop a low-cost, user-friendly system that utilizes Arduino and OpenCV to recognize hand gestures accurately and in real-time. The system should be able to identify a range of hand gestures, including basic gestures such as thumbs up, thumbs down, and pointing, as well as more complex gestures such as hand shapes used in sign language.

The problem statement for this project is to design, develop, and evaluate a reliable hand gesture recognition system using Arduino and OpenCV that can accurately and quickly recognize a range of hand gestures in real-time for various practical applications.

2. PROJECT REQUIREMENTS SPECIFICATION

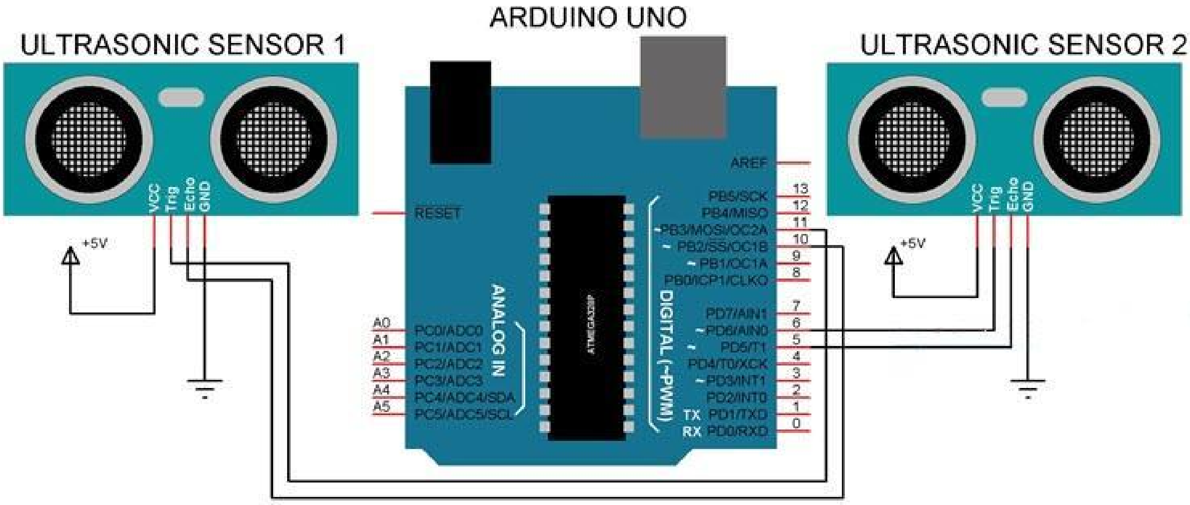
Hardware

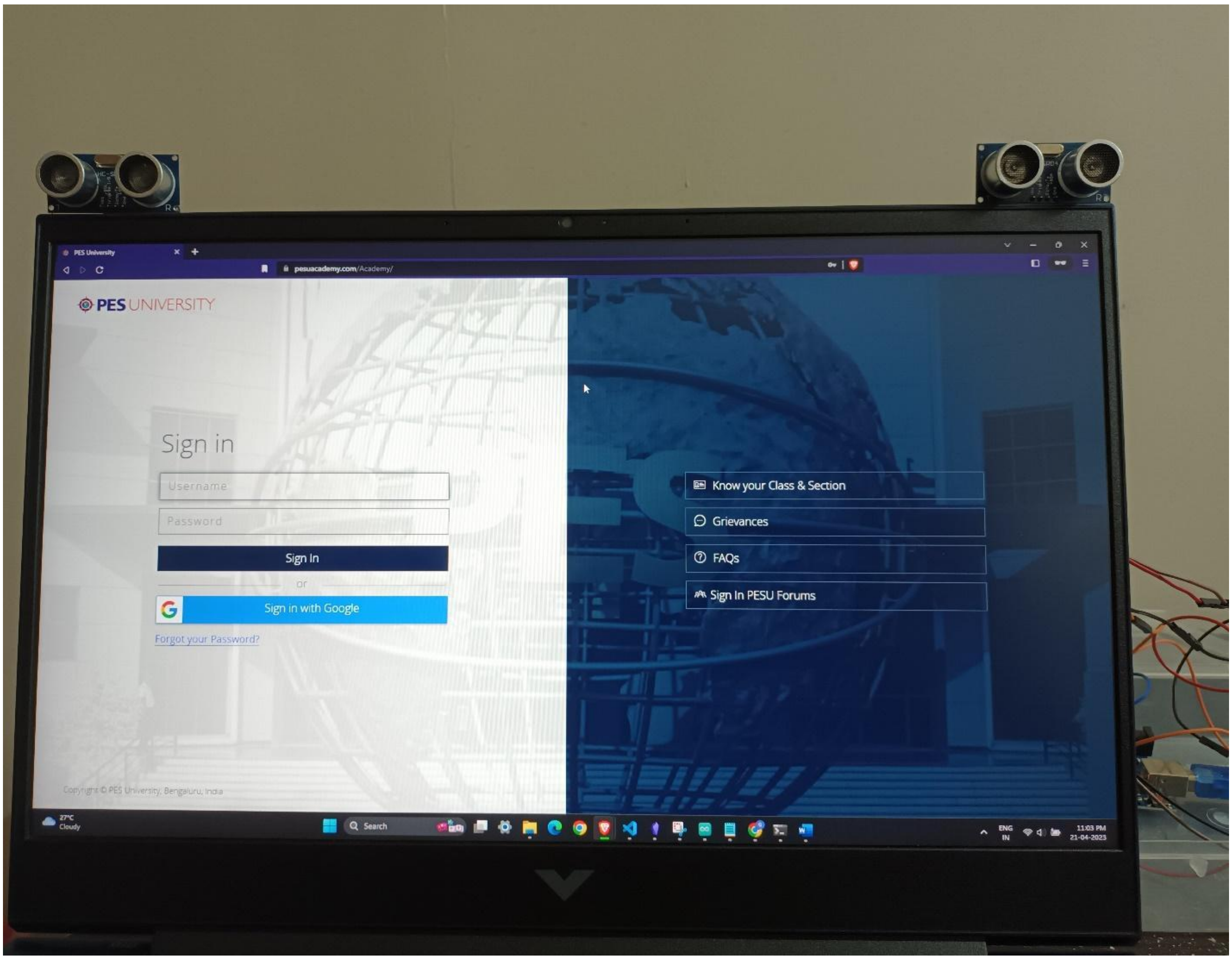
- Arduino UNO Board
- Ultrasound Sensors
- Wires

Software

- Arduino IDE
- Python Libraries
    - PyAutoGUI
    - OpenCV
    - MediaPipe
    - Tensorflow

3. PROJECT DESIGN





- The first image shows the circuit diagram that was used to set up the hardware components required for the project.
- The second image shows the arrangement of the ultrasound sensors on the computer system being used, with the Arduino UNO visible in the background.
- The ultrasound sensors in the circuit diagram are placed on either end of the laptop screen, and attached using double-sided tape.
- The idea is to set certain computer events based on gestures recognized by OpenCV, or based on a combination of gestures picked up by the ultrasound sensors and OpenCV.
- The code uploaded to Arduino works in such a way that particular gestures made in front of the ultrasound sensors will print certain strings to the serial port.
- PyAutoGUI is used to trigger certain events by programmatically controlling the keyboard based on the string that gets printed to the serial port.
- At the same time, hand signs that are picked up by OpenCV functions will cause PyAutoGUI to press a separate bunch of keys.

- The following gestures in front of the ultrasound sensors will do the mentioned tasks:
    - Place palm in front of the right ultrasound sensor - Presses the 'tab' key.
    - Place palm in front of the left ultrasound sensor - Presses the 'caps lock' key.

- The following gestures in front of the web camera will do the mentioned tasks:
    - "Fist" sign - Holds down 'ctrl' key.
    - "Peace" sign - Holds down 'alt' key.

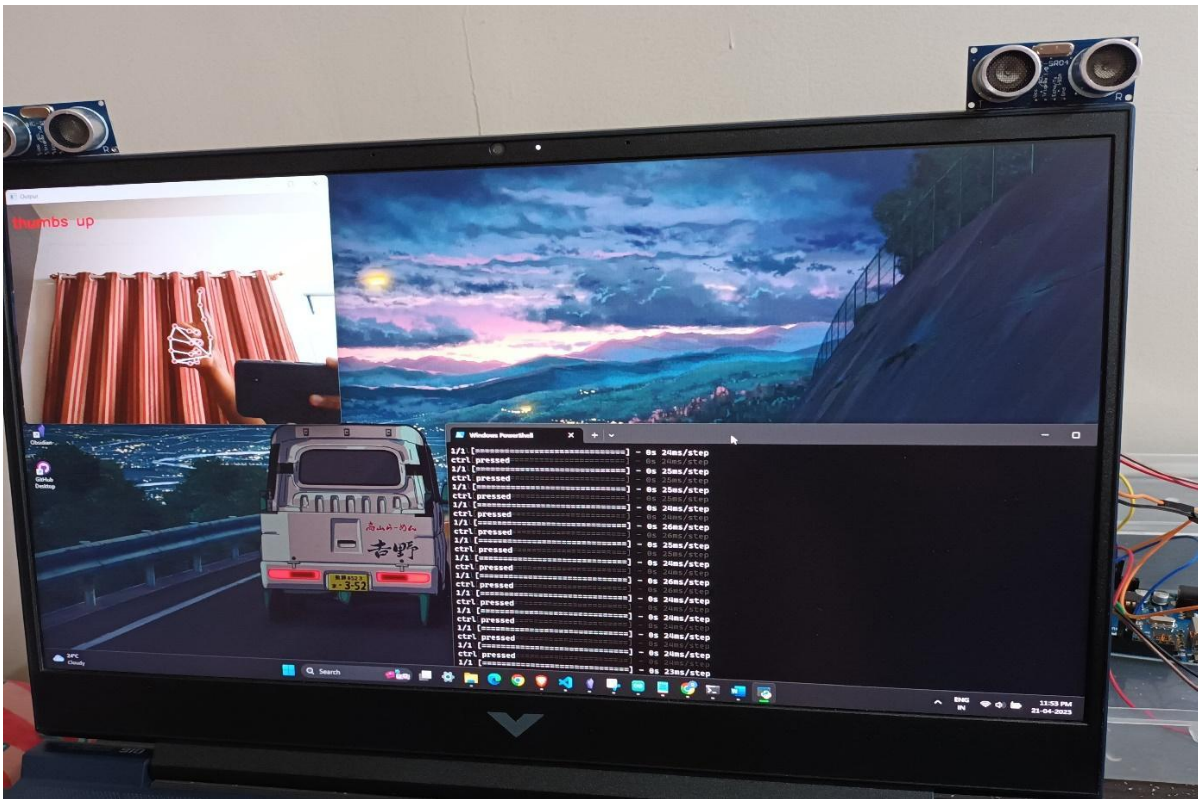
    - “Stop” sign - Releases the ‘ctrl’ key or the ‘alt’ key, based on which one of them is currently being held down.
    - “Okay” sign - Presses the “Enter” key.
    - “Thumbs up” sign - Scrolls up in the current page.
    - “Thumbs down” sign - Scrolls down in the current page.

- Using these principles, a key or a combination of keys can be programmatically pressed to trigger certain events.

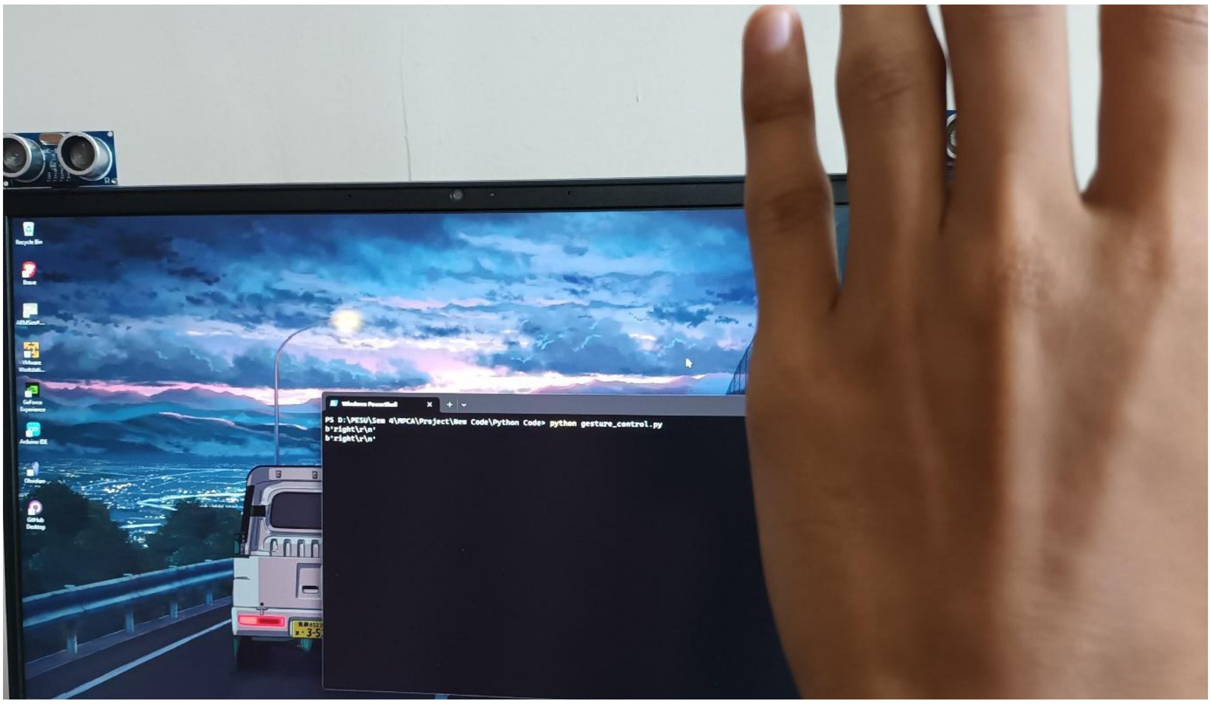
4. OUTPUT RESULT

- As an example, the “fist” gesture in front of the web camera is picked up by OpenCV, and leads to PyAutoGUI holding down the “ctrl” button.
- At the same time, swiping one’s hand in front of the right ultrasound sensor leads to PyAutoGUI pressing the “tab” key.
- When these two actions are done in tandem, it allows us to switch the current tab in a browser, for example.
- Similarly, single gestures detected by OpenCV are also used for other tasks.
- For example, showing the “thumbs up” gesture in front of the web camera scrolls up the current page.

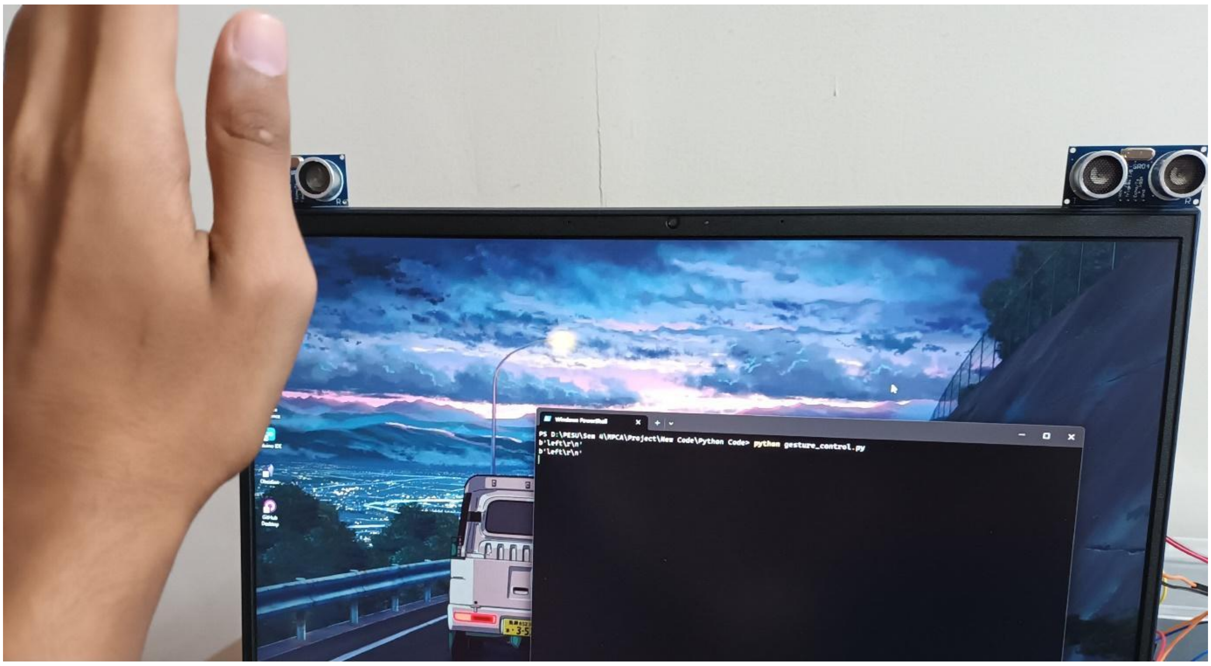
- The first image shows OpenCV detecting a “thumbs up” gesture. This is done by means of the web camera that is part of the PC.
- The second image shows that the fact that our hand is in front of the right sensor is being detected (as seen in the command terminal).
- Similarly, the third image shows the detection of our hand, which is in front of the left sensor.
- All these three lead to keyboard key presses, a combination of which can trigger an event.



```
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 25ms/step
ctrl pressed
1/1 [==============================] - 0s 25ms/step
ctrl pressed
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 26ms/step
ctrl pressed
1/1 [==============================] - 0s 25ms/step
ctrl pressed
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 26ms/step
ctrl pressed
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 24ms/step
ctrl pressed
1/1 [==============================] - 0s 23ms/step
```



```
PS D:\PESU\Sem 4\MPCA\Project\New Code\Python Code> python gesture_control.py
b'right\r\n'
b'right\r\n'
```



```
PS D:\PESU\Sem 4\MPCA\Project\New Code\Python Code> python gesture_control.py
b'left\r\n'
b'left\r\n'
```