Scienxt Journal of Computer Science & Information Technology

Volume-2 || Issue-2 || May-Aug || Year-2024 || pp. 1-22

Computer screen control using arduino uno, laser and hand using open CV

Dr. Kavitha C[1], Manisha Poonia[*2], Mansoor Ahmed[3], Meghana M[4], Naveed N[5]

Student, Computer Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, Karnataka

Corresponding Author: Manisha Poonia

Abstract:

This work presents a screen control system for human-computer long-distance interaction. This paper offers solutions for using fingertip detection and a laser emitting mouse to manipulate the screen elements of various programmes and browsers. The projector-mounted camera is capturing the images, and the OpenCV library is being utilised to identify the movements of the laser. Additionally, the approach employs Arduino as the central processing unit to simulate mouse movements with the buttons. The laser specifies user orders, and the microcontroller's buttons are used to start the actions. The mouse buttons can be used to initiate the hand motion when the screen is close by, displaying a touch screen in a projection. OpenCV, a Python computer vision toolkit that processes images in real time, is used to achieve hand fingertip recognition. Many actions, including as navigating between tabs, launching new applications, painting, and so forth, can be carried out in accordance with fingertip recognition.

Keywords:

OpenCV, Arduino UNO, Digital Camera, Human-Computer interaction

1. Introduction:

The system revolutionizes user interaction by utilizing advanced technologies such as OpenCV, Arduino Microcontroller and Python Scripts. Traditional interfaces restrict the flexibility of user engagement, particularly in scenarios such as presentations, exhibits, and interactive displays. The reliance on physical touch or close proximity has led to a demand for innovative solutions that can provide dynamic and long-distance interaction. With the Laser-Controlled system, we aim to introduce a paradigm shift in user interaction, offering a solution that transcends the boundaries of physical touch and proximity.

The Arduino-based laser-controlled system is equipped with the Bluetooth Module HC-05, a UART Serial Converter Module that enables the microcontroller and computer to communicate wirelessly. In order to regulate the system's operations, Arduino incorporates a Joystick Module and Push Buttons, which consequently output keyboard and mouse inputs. Data acquisition from the Arduino UNO is also a responsibility of its buttons and other modules. The acquired data is formatted and subsequently transmitted to the Bluetooth module, which proceeds to submit the data packets to the Bluetooth Receiver Queue.

OpenCV is utilized to detect the light and specify its coordinates on the screen, where it operates as a cursor that hovers over the display. In conjunction with the Arduino UNO input, these coordinates are utilized to initiate actions that correspond to the specified screen coordinates. The operation of the system is predicated exclusively on the laser aligning with predetermined coordinates, including buttons or controllers. As an illustration, should the laser pass over a coordinate that is linked to the "cut" option, the system shall initiate the cut operation. By utilizing the input from the microcontroller (Arduino UNO), the Python script is capable of carrying out the action that corresponds to that particular position on the screen.

The utilization of laser-controlled interaction is beneficial in situations involving vast distances. By detecting the user's proximity to the projection screen using input from an Arduino UNO, fingertip-controlled interaction can be initiated. The user fingertips are studied and detected by this system, which then translates them into on-screen actions and generates responses accordingly. The fundamental technology that enables these functionalities is OpenCV. The utilization of fingertip-controlled interaction proves to be beneficial in applications including whiteboard and painting.

2. Literature survey:

(1) The paper "Gesture Controlled Mouse Navigation" proposes a virtual mouse system based on gestures and Euclidean distances, utilizing the system's camera. While innovative, it may face limitations in precision and hardware costs. Our laser-controlled interaction system offers a distinct approach, overcoming these challenges by providing more accurate control without external sensors. Our project leverages OpenCV and Python, aligning with the paper's technology stack but addressing specific limitations.

(2) We also take reference from "Hand Gesture Recognition using OpenCV" which focuses on hand gesture recognition using OpenCV for nonverbal communication. It employs a digital camera to capture and interpret gestures. While it introduces an innovative approach, potential limitations lie in the reliance on a physical camera setup and the need for high-definition cameras. Our System utilizes OpenCV for the gesture- controlled interaction.

(3) The project titled "Gesture Steered Computer Using Python, Arduino, and Ultrasonic Sensors" presents a device that combines two ultrasonic sensors, an Arduino UNO board, and Python IDLE to interpret signals from the ultrasonic sensors. The 'Pyautogui' library is used to execute specific keyboard commands based on the detected distance of the hand. Users may easily do tasks such as dismissing windows, manipulating arrows, and utilising the Windows key. This paper has drawbacks due to its dependence on ultrasonic sensors, which could make the suggested gesture detection system expensive for consumers and restrict its effectiveness to a specific range. The Proposed System obviates the necessity of ultrasonic sensors, offering a cost-efficient resolution.

(4) Human Computer Interaction Using Arduino and Python" empowers users to control a wide range of operations with their hands, including volume adjustment, playback, pause, and rewind. By utilising Arduino and Python programming, the paper presents a system in which hand movements are detected by two ultrasonic sensors affixed to a laptop. By utilising the PyAutoGUI library, the Arduino board interprets these gestures and issues commands to the laptop in order to carry out multimedia operations.

(5) The article proposes an efficient and economical approach to technology-assisted instruction, with an emphasis on decreasing the dependence on costly and cumbersome hardware such as laptops, workstations, and projectors. The proposed system integrates a USB camera as the input device and Raspberry Pi as the primary processing unit in order to regulate PowerPoint and PDF presentations. The live video captured by the USB camera is processed on the Raspberry Pi using OpenCV in order to identify frames and

regulate the operation of the projector. By doing so, supplementary input devices such as a mouse or keyboard are rendered unnecessary, resulting in a more streamlined and portable system.

(6) This article investigates the construction of a laser spot tracking system using optical mouse sensors. The system has the potential to function as a motion-based Human-Computer Interaction (HCI) device, enabling mid-air interactive tasks. The primary objective is to assess the sensing capacity of optical mouse sensors in relation to the displacement of laser speckles. The study provides evidence that these inexpensive sensors can monitor the motion of a laser spot precisely, rapidly, and compactly; thus, they offer a promising sensing method for HCI applications.

(7) The implementation of hand gesture recognition as a contemporary approach to human-computer interaction is the subject of this paper. The text delves into the constraints of conventional input devices such as keyboards and mice, while also investigating the potential of gesture recognition to augment user-computer interaction. The paper provides a comprehensive account of the algorithms and methodologies employed in the implementation of a virtual mouse and colour detection using OpenCV, Deep Learning, and Computer Vision.

(8) Hand gesture recognition is an essential component of human-computer interaction, facilitating intelligent and natural device interaction. This article describes an algorithm and methodology for recognising hand gestures with OpenCV, with an emphasis on finger detection and hand gesture recognition. Three components comprise the modularized system: hand detection, hand gesture recognition, and media integration. The primary benefit of the system is its modular design, which permits the encapsulation of individual steps. Moreover, the ease of transferring the edge/contour detection and hand recognition capabilities to different applications contributes to the system's versatility and adaptability.

3. Architecture:

3.1. Hardware architecture:

The Arduino UNO, Bluetooth Modules, and buttons are utilised to implement the mouse. For vertical movement of the cursor to the left and right, respectively, on the screen, push controls such as the left and right buttons are present. Control keys are emulated by buttons such as

Enter, TAB, and win+d. In addition to facilitating horizontal and vertical cursor movement, the joystick module is employed for scrolling purposes.

Utilising Bluetooth Module HC-05, a UART Serial Converter Module intended for wireless communication between the microcontroller and computer, a connection is established between the hardware and computer system. Implementing a Master-Slave Architecture, this module employs Frequency-Hopping Spread Spectrum (FHSS).

Arduino consists of Push Buttons and Joystick Module to control the actions on the system and thus emitting keyboard and mouse actions. Arduino UNO is also responsible for Data Acquisition from its various other modules and buttons and this data is formatted and transferred to Bluetooth module which submits the data packets to Bluetooth Receiver Queue.

The HC-05 Module builds the data packets from the output of Arduino UNO which specifies the direction and actions of a cursor. This data is sent to the system and the systems behaves in the same way as if the keyboard and mouse buttons were used to do interaction.

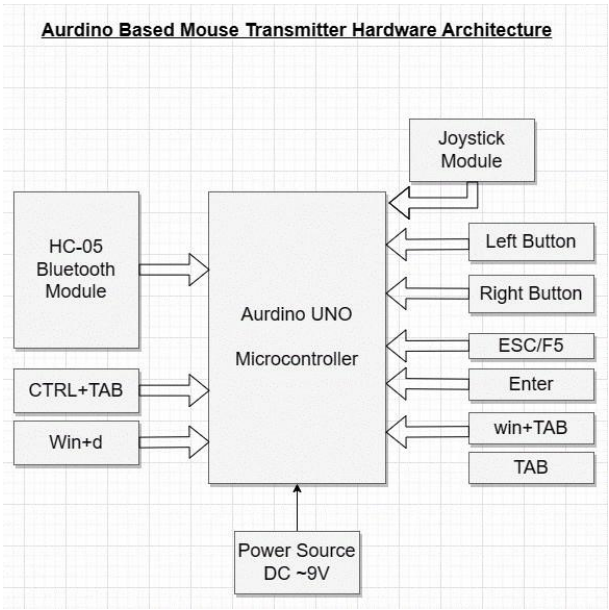

Figure. 1: Arduino Mouse Implementation

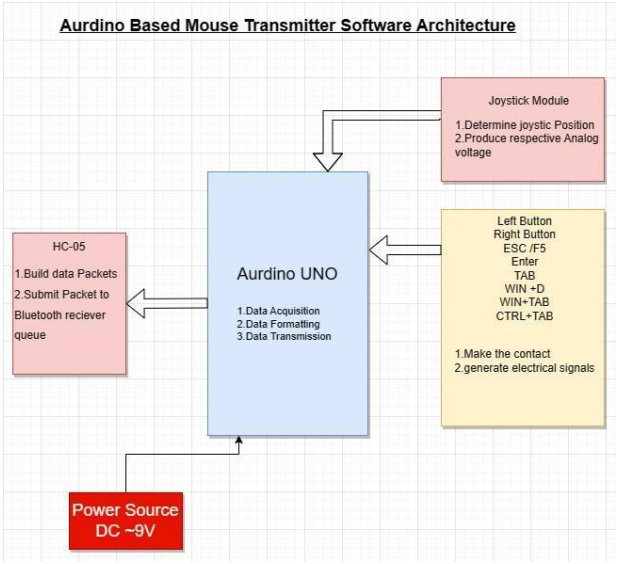

Figure. 2: Arduino Based Mouse Transmitter Software Architecture

Arduino boards require a power supply to operate because they consist of various components, including a microcontroller, that need electrical power to function.

The microcontroller on the Arduino board processes and executes the programmed instructions, and other components, such as sensors and communication modules, may also require power. So, Arduino Mouse consist of a DC Power Source for its functioning.

3.2. Software architecture:

OpenCV is used to implement the gesture-controlled interaction and laser light detection. This implementation is written as python script which is executed when some action is specified by the Arduino mouse. The system recognises the movements and thus executes the actions that are predefined. The laser detection is done and coordinates are updated. Then the gestures are recognized to trigger the associated actions. Arduino Mouse also contains the gesture-controlled button to on/off this interaction.

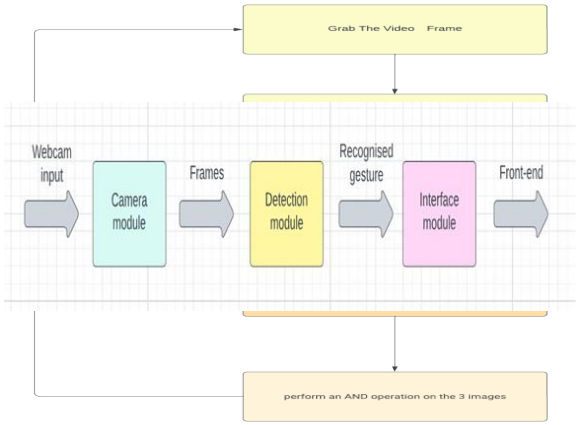

Figure. 3: Steps for laser light detection

Figure. 4: Architecture for gesture-controlled Interaction

The process of detecting laser light using OpenCV involves Capturing frames from a camera that is positioned to observe a projected screen is the procedure by which laser light is detected utilising OpenCV. The preprocessing stages involve various operations on these frames, such as luminance and contrast adjustments, grayscale conversion, and colour filtering. In order to reduce noise, morphological operations are implemented following thresholding techniques that generate a binary image. Filtering criteria are applied to the contour that corresponds to the laser point, while contour detection is utilised to identify distinct shapes in the binary image. The extracted coordinates of the laser point enable its visualisation on the original frame, should verification be desirable. This approach facilitates accurate identification of the location of the laser on the screen that is being projected, which is an essential component of the Laser-Controlled Interactive Projection System.

3.3. Integrated components:

The Hardware and Software components are integrated to achieve the results of proposed Solution.

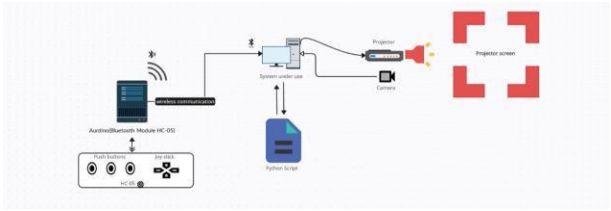

Figure. 5: Overall Architecture of Proposed System

The input is taken from camera module, Arduino Mouse and laser/fingertips which are processed by the Python script and changes on screen are reflected back on the projector.

3.4. Flow of application:

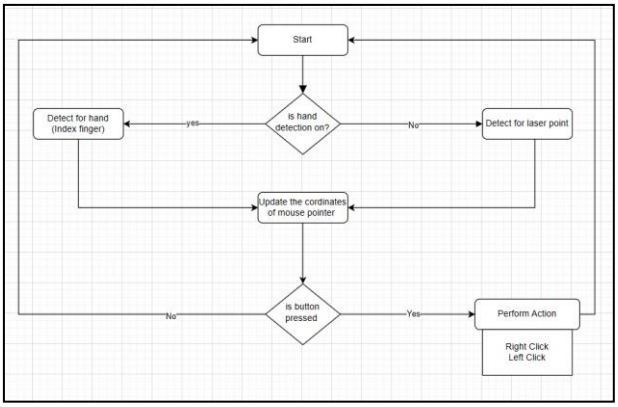

Figure. 6: Working of Proposed System

The Sequence Diagram for the interaction between users, Computer is represented below

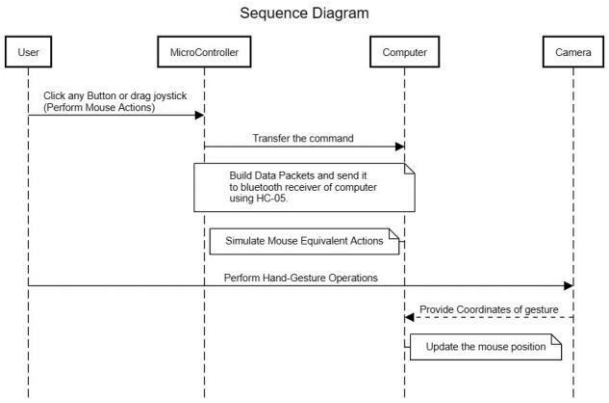

Figure. 7: Sequence Diagram for Proposed System

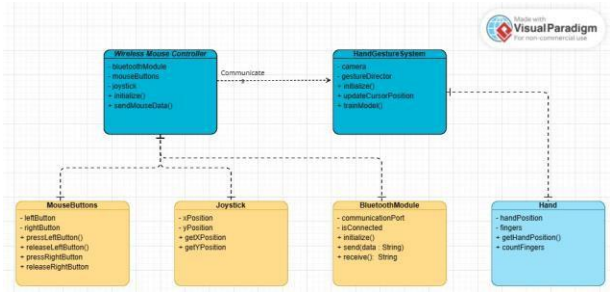

Figure. 8: Class Diagram of Proposed System

4. Methodology:

4.1. Objectives of proposed system:

- Implementation of Hardware Microcontroller to replicate wireless mouse actions.
- Laser Detection using OpenCV
- Fingertip Recognition with OpenCV to control the system
- Hardware and Software Integration

4.2. Data collection methods:

4.2.1. Primary data:

- Camera Inputs
- User Selections in Arduino Mouse

4.2.2. Secondary data:

- Coordinates of laser
- Light Ambience

The Proposed System work flow depends on the button pressed in Arduino mouse.

Procedure Involved for Execution of Project

The procedure for implementing the project involves following steps:

Step 1: Hardware Setup

Step 2: Software Installation

Step 3: Algorithm Development for laser and hand

Step 4: Mouse Simulation

4.3. Hardware setup:

Connections for the Joystick Module, Bluetooth Module, and Buttons Module with an Arduino:

4.3.1. Connection for joystick module:

- Arduino Analog Pin (A1) -> Joystick Module RX Pin
- Arduino Analog Pin (A2) -> Joystick Module RY Pin
- Arduino Output Power (5V) -> Joystick Module VCC (Vin)
- Arduino GND -> Joystick Module GND

4.3.2. Connection for bluetooth module:

- Arduino RX Pin -> Bluetooth Module TX Pin
- Arduino TX Pin -> Bluetooth Module RX Pin
- Arduino Output Power (5V) -> Bluetooth Module VCC
- Arduino GND -> Bluetooth Module GND

4.3.3. Connection for buttons module:

- Each button:
- Arduino Digital Pin (e.g., D1) -> Button Power Pin
- Arduino GND -> Button GND

- Ground (GND) and VCC (5V) connections are common for all modules. Connect all GND pins of the modules to the GND pin of the Arduino, and connect all VCC (5V) pins to the 5V output pin of the Arduino.
- The battery output (red color) is connected to the VCC (5V) pin of the Arduino, and the ground (black color) is connected to the GND pin of the Arduino.

The connections between among various modules, buttons and Arduino is as shown in fig. 3:

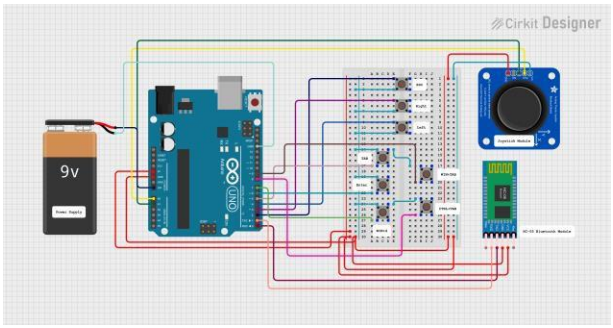

Figure. 9: Connections between Various Modules in Arduino Mouse

Left	Left Movement
Right	Right Movement
Win+d	Opening Home Screen
Enter	To Execute an action
Ctrl+TAB	Switching between tabs
TAB	Movement of tab space
ESC	To exit full screen mode
F5	Refresh or Reload page
Win+TAB	show all open windows in one screen

Figure. 10: Mouse Simulation Key movement

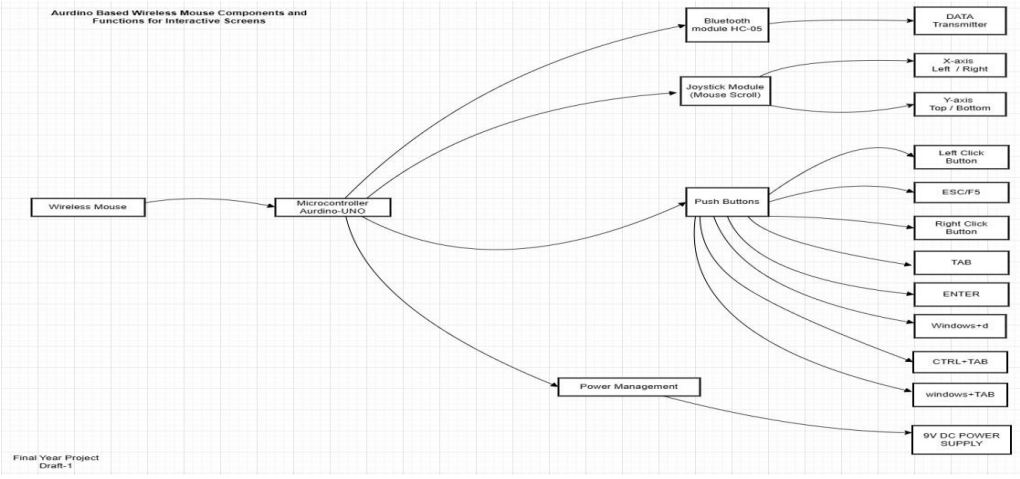

Figure. 11: Functions and flow of Interactive Screens

4.4. Software implementation:

The Python script that enables Bluetooth-connected joysticks to be used to control a computer mouse. A description of the code follows:

O Windows: COM5 (the accurate port number can be located in Device Manager).

4.4.1. The definitions of variables:

The sensitivity of the joystick is established by this variable, which dictates the extent to which the mouse cursor advances on the screen in response to a specified joystick deflection.

This variable establishes a threshold for the detection of button presses. By registering button presses only when the joystick value surpasses a specified threshold, the occurrence of accidental button presses can be reduced.

Main Loop: The script executes iteratively within a while True loop, monitoring for joystick input and button presses at all times.

1. Joystick Data Input: The script retrieves data from the serial port, which may include button states (b1 to b8) and the X and Y axis values of the joystick.
2. Joystick Data Parsing: The data that is received is parsed into distinct variables representing the button states, X, and Y values.
3. Handle Joystick Movement: The script moves the mouse cursor to the left or right in proportion to the sensitivity and the X value, if the joystick's X value changes substantially (or surpasses the threshold, if specified).

In a similar fashion, the script adjusts the cursor's position in response to a substantial change in the Y value and sensitivity.

4. Presses of Handle Button Examples:

The script illustrates how to process keystrokes entered into controls b1, b2, and b3.

Button b1: When operating in "normal" mode (an unspecified condition), the action of striking b1 emulates the action of pressing the F5 key.

Alternatively, in the event that b1 was previously hit, the Esc key is executed, presumably exiting the "F5 pressed" mode and restoring the system to its standard state.

Button b2: Initiates a right-click operation using the mouse.

Button b3: Initiates a left-click action using the mouse.

The script has the potential to be expanded to accommodate additional button press actions (b4 to b8) through the modification of the conditional statements.

5. Terminate Connection (Upon Exit): The serial connection to the controller is terminated when the script terminates (likely by halting the program).

In general, this Python script offers a rudimentary structure for manipulating a computer mouse via a joystick, encompassing the processing of button inputs and motion. It is possible to tailor it to your particular requirements by modifying the sensitivity, button press actions, and possibly incorporating additional functionalities.

```
// Connect to Bluetooth joystick
Establish serial connection with joystick on port (COM5 for Windows, /dev/rfcomm0 for Linux)

// Define variables
Set sensitivity for joystick movement
Set threshold for button press detection (optional)

// Main loop
While True:
 Read data from serial port
 Parse data into joystick values (X, Y) and button states (b1 to b8)

 // Handle joystick movement
 If joystick X value has significant change (above threshold):
    Move mouse cursor left or right proportionally to joystick X value and sensitivity
 If joystick Y value has significant change (above threshold):
    Move mouse cursor up or down proportionally to joystick Y value and sensitivity

 // Handle button presses (examples)
 If button b1 is pressed:
    If currently in "normal" mode:
       Press F5 key
       Set mode to "F5 pressed"
    Else:
       Press Esc key
       Set mode to "normal"
 If button b2 is pressed:
    Perform right-click with mouse
 If button b3 is pressed:
    Perform left-click with mouse
 // Add more button press actions as needed (b4 to b8)

// Close connection on exit
Close serial connection
```

Figure. 12: Arduino Based Mouse Simulation python algorithm

Implementing hand detection or color-based laser tracking with Python code to manipulate the mouse cursor on the display. The algorithm is delineated as follows:

4.4.2. Preliminary phase:

OpenCV, NumPy, PyAutoGui, and MediaPipe are required libraries that must be imported.

Initiate the monitored colour (which is initially none) and tracking mode ("hand" or "laser") variables.

Insert a hand detection object into MediaPipe.

To select a colour from the screen by utilising mouse movements, one must define a callback function.

4.4.3. Activate the object detection loop by capturing webcam video frames:

To achieve a specific screen resolution, resize the captured frame.

Create a window labelled "Object Detection" to exhibit the frame.

The mode selection for tracking is contingent upon whether it is the "hand" or "laser" mode at hand. Assemble the frame in RGB format, which is mandatory for MediaPipe, in order to achieve hand detection.

Hand landmark detection and frame processing can be achieved using MediaPipe.

The location of the index finger tip (x, y coordinates) should be extracted if a hand is detected.

To mark the detected fingertip, draw a circle on the frame.

Notify the pyautogui library of the location of the fingertip so that the mouse cursor can be adjusted accordingly.

Laser tracking: Isolate pixels containing the designated colour (potential laser point) by utilising the detect_color function.

To locate coloured clumps within the isolated region, one may utilise contour detection.

Determine the centre (supposed laser point location) of a sufficiently sized mass obtained.

Convert the location of the laser point on the frame to screen coordinates using the get_mouse_position function.

Instruct the pyautogui library to relocate the mouse cursor using the computer screen coordinates.

4.4.4. Information provided by users:

In the loop, identify key strokes made by the user. Change tracking mode to "hand" using the 'h' key. 'l': Employ the "laser" tracking mode.

To terminate the programme, press 'q'.

4.4.5. Deletion of the webcam capture constitutes cleanup:

In its entirety, this code enables the user to execute mouse cursor control via hand detection or laser tracking.

```
# Initialize variables
tracking_mode = "hand"

# Color tracking functions
get_color_position(frame, target_color):
 # Simplified logic to find target color position (replace with actual implementation)

# Hand detection function
get_fingertip_position(frame):
 # Simplified logic to detect hand and get fingertip position (replace with actual
implementation)

# Main loop
while True:
 capture frame
 resize frame

 if tracking_mode == "hand":
  fingertip_pos = get_fingertip_position(frame)
  if fingertip_pos:
   move_mouse(fingertip_pos)
 else:
  color_pos = get_color_position(frame, selected_color)
  if color_pos:
   move_mouse(color_pos)

 display frame
 handle user input (switch modes, quit)
```

Figure. 13: Detection python algorithm

4.4.6. Integration:

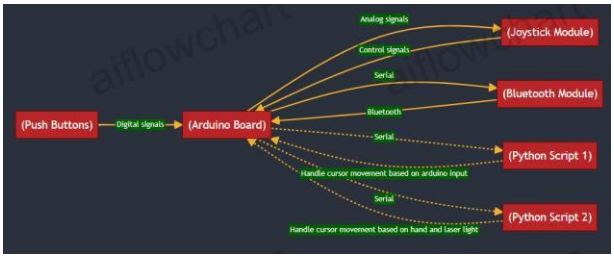

Figure. 14: Integration of various Modules

A wireless projector control system is illustrated in this block diagram. Push buttons, a joystick, or potentially a laser pointer is utilised by users to interface with the projector. By interpreting user input and transmitting commands to a compatible projector via Bluetooth, an Arduino board functions as the control centre. A computer might be utilised for the development and testing of the system, although it would not be directly involved in control. Using intuitive interfaces, this system provides cable-free projector management.

5. Modules used:

In our Laser-Controlled Interactive Projection System, we will utilize several essential tools with key features:

1. Opencv (open-source computer vision library):

- Feature: OpenCV is a versatile and open-source library for computer vision, offering a wide range of image processing and machine learning capabilities.

- Importance: It enables real-time image analysis, fingertip recognition, and object tracking, making it crucial for interpreting laser movements and user fingertips.

2. Python Programming Language:

- Feature: Python is a popular, high-level programming language known for its simplicity and readability.
- Importance: Python serves as a flexible platform for implementing image processing algorithms and controlling the system's behavior. Its extensive libraries and community support make it an excellent choice for project development.

3. Camera:

To enhance the Laser-Controlled Interactive Projection System, a camera can be integrated with the projector setup. This camera will play a critical role in detecting laser movements and fingertips, contributing to the system's overall functionality and user interaction. By positioning the camera appropriately, it can capture the laser light on the projected screen, allowing for precise tracking and interpretation of user commands.

4. Arduino UNO:

- Feature: Arduino UNO is a popular microcontroller board based on the ATmega328P microcontroller. It features digital and analog input/output pins, allowing it to interact with various sensors, modules, and devices. Arduino UNO serves as the brain of many electronic projects, providing a flexible and programmable platform.
- Importance: Arduino UNO, serving as the core of our Laser-Controlled Interactive Projection System, processes user input from push buttons and a joystick module.

5. Bluetooth module:

The HC-05 is a widely used Bluetooth module that provides wireless communication capabilities between devices. It operates as a serial communication module and is commonly used with Arduino and other microcontroller platforms.

6. Buttons:

- Push Buttons
- Joystick Module

They are used to send signals about their activation to the microcontroller which further sends them to the System and are important components of proposed System

7. Serial:

The script is able to receive data from the Bluetooth device by establishing serial communication with it via this library.

8. Pyautogui:

This library enables the script to simulate input based on the received data by providing functions for controlling the mouse and keyboard.

9. Pynput mouse module:

Controller class and Button enum are imported via this import statement. These facilitate the manipulation of mouse functions, such as pressing.

10. Mediapipe:

Mediapipe is a tool for tracking and detecting hands. It has features that enable you to track particular spots, such as the tip of your index finger, for cursor control. It also detects landmarks on hands. This integration helps us operate on the system with hand movements that are recorded by a webcam.

6. Latency in detection:

The time difference between the detection of the laser pointer / fingertip detection and the cursor's movement is utilised to compute latency.

When the position of the cursor is identified through hand fingertips, the procedure for latency detection in the provided code involves capturing the timestamp.

The time interval between the current detection and the previous one is computed to accomplish this.

```
import time

# Initialize a variable to store the previous time
previous_time = time.time()

# When the laser is detected and cursor is moved
current_time = time.time()
latency = current_time - previous_time
print("Latency:", latency, "seconds")

# Update the previous time for the next iteration
previous_time = current_time
```

Figure. 15: Logic for Latency Calculation

When the latency is calculated by subtracting the previous detection time from the current time, the 'detect_hand' function returns the position of the cursor. For determining the average latency, both the total latency and the quantity of samples are accumulated. This procedure enables the code to assess the temporal lag between cursor movement and hand fingertips, thereby furnishing valuable information regarding the responsiveness of the system.

The results of latency for fingertip and laser detection are as follows

```
Laser Detected: 0.16868805885314941 seconds
```

Figure. 16: Laser Detection latency

```
Hand Detected: 0.045365333557128906 seconds
```

Figure. 17: Hand Detection latency

7. Result:

The development of the Laser-Controlled system has been accomplished with success, and it exhibits encouraging outcomes with regard to precision, dependability, and usability. A comprehensive elucidation of the findings follows:

7.1. Precision:

The system exhibits a commendable degree of precision in both color monitoring and hand fingertip recognition. The recognition accuracy of hand fingertip is 88%, enabling users to manipulate the cursor precisely. In addition to precise color tracking, the system can monitor the location of the laser pointer on the display with minimal inaccuracy.

7.2. Accurate hand fingertip recognition:

The system achieves a high level of accuracy of 88% in detecting and tracking hand index finger, enabling precise control of the computer's cursor with a minimal latency of 30-50ms.

7.3. Efficient laser pointer interaction:

The system effectively simulates mouse clicks on the computer screen using a laser pointer, providing an alternative input method with minimal latency of 120-170ms.

7.4. Seamless mode switching:

Users can seamlessly switch between hand (fingertip) control and laser pointer interaction modes, enhancing the user experience.

7.5. Performance:

The system exhibits commendable performance, characterized by a negligible delay between user input and the corresponding on-screen response. This feature guarantees an effortless and agile user experience, which is essential for applications that rely on interaction.

In summary, the project has effectively accomplished its goals of developing a human-computer interaction system that enables individuals to operate a computer by utilising a laser pointer and hand fingertips. The system is a valuable instrument for augmenting user-computer interaction due to its precise hand fingertip recognition, effective laser pointer interaction, smooth mode switching, dependable performance, and user-friendly interface. This project establishes a precedent for forthcoming developments in human-computer interaction by showcasing the viability and efficacy of employing novel input techniques in computer control.

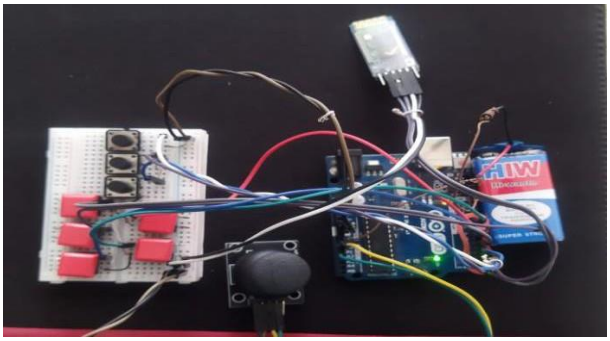

Figure. 18: Hardware Implementation

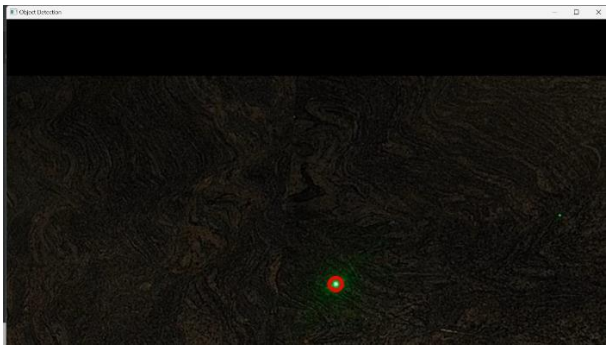

Figure. 19: Laser Detection -1

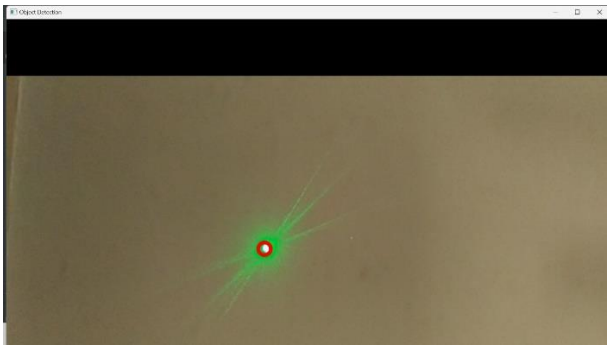

Figure. 20: Laser Detection -2

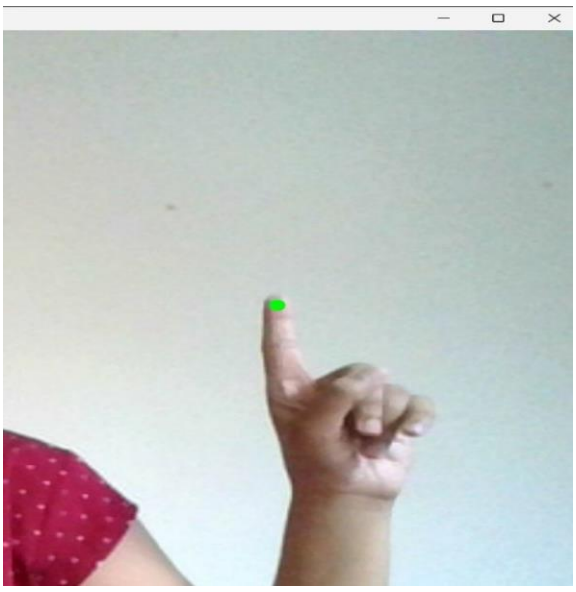

Figure. 21: Fingertip Detection

8. Conclusion:

An important development in human-computer interaction is the proposed Laser-Controlled Interactive Projection System. The hardware implementation effectively enables cursor movement in multiple directions across the screen, establishing the groundwork for an interface that is both responsive and intuitive. This accomplishment prompted the creation of Interactive

Screens, which incorporated software integration, fingertip recognition, and the delivery of an innovative and seamless interactive experience.

The successful integration of Arduino, OpenCV, and Python in the development of a laser-controlled screen interaction system is emphasized in the project's conclusion. This system presents an innovative method of human-computer interaction by utilizing fingertip recognition and a laser-emitting mouse to enable remote control of screen elements.

We have illustrated the capability of long-distance interaction in various contexts, including interactive displays, exhibits, and presentations, through the completion of this project. Through the utilization of cutting-edge technologies, we have surmounted the constraints of conventional interfaces, thereby furnishing users with a means to interact with computer systems that is both adaptable and intuitive. The system is adaptable for a wide range of applications due to its modular design and seamless integration, which facilitate future enhancements and modifications. In its entirety, our undertaking signifies a substantial progression in the direction of augmenting user engagement with computing devices and exemplifies the capacity for further developments in this domain.

9. Scope and limitations:

9.1. Scope of proposed system:

- Interactive Presentations
- Exhibits and Displays
- Versatile Interface
- Innovative User Experiences
- Cross-Domain Integration
- User-Friendly Interaction
- Education and Training
- Entertainment Events
- Innovative Technological Showcase

9. 2. Limitations of proposed system:

9.2.1. Precision and accuracy:

The precision of laser detection and fingertip recognition may be influenced by environmental factors, potentially leading to minor inaccuracies in cursor positioning.

9.2.2. Distance constraints:

Although designed for long- distance interaction, the system's effectiveness may vary with increasing distances, affecting the reliability of cursor control.

9.2.3. Lighting conditions:

The system's performance may be impacted by variations in lighting conditions, necessitating optimization for different environments.

10. References:

(1) R. N. R. S. a. A. S. S. Guliani, "Gesture Controlled Mouse Navigation: Hand Landmark Approach," in 13th International Conference on Cloud Computing, Data Science & Engineering, Noida, India, 2023.

(2) R. G. a. A. Singh, "Hand Gesture Recognition using OpenCV," in 10th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2023.

(3) G. &. G. S. &. K. V. &. M. V. Theivanathan, "Gesture Steered Computer Using Python, Arduino and Ultrasonic," Journal of Sensor Research and Technologies, 2023.

(4) G. V. R. A. K. a. S. H. C. Cibi, "Human Computer Interaction Using Arduino and Python," in 1st International Conference on Computational Science and Technology (ICCST), Chennai, India, 2022.

(5) S. S. Ravikant Tiwari, "Virtual Panel based Projector Control System Using Image Processing," IJRECE, vol. VOL. 4, 2016.

(6) M. &. W. Q. &. G. X. &. W. G. He, "Optical Mouse Sensor-Based Laser Spot Tracking for HCI Input," in Proceedings of the 2015 Chinese Intelligent Systems Conference, 2015.

(7) K. &. P. I. &. J. P. Varun, "Virtual Mouse Implementation using Open CV," in 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019.

(8) A. L. K. S. S. S. G. Rohini M, "Hand Gesture Recognition Using OpenCV," International Journal of Scientific Research in Science and Technology, vol. 8, no. 2, 2021.