Voice and Gesture-Driven IoT System: A Novel Approach for Enhanced HCI

Vinay Bodem
dept. of Artificial intelligence and data science(AIDS)
GMRIT
Rajam, India
bodemvinay@gmail.com

Lakshmi Devi. N
dept. of Computer Science and Engineering(AIML)
GMRIT
Rajam, India
lakshmidevi.n@gmail.com

Tulasi Ram Veerni
dept. of Artificial intelligence and data science(AIDS)
GMRIT
Rajam, India
Tulasiramveerni007@gmail.com

Yamini Ponduru
dept. of Artificial intelligence and data science(AIDS)
GMRIT
Rajam, India
yaminiponduru2024@gmail.com

Anil Kumar Koduru
dept. of Artificial intelligence and data science(AIDS)
GMRIT
Rajam, India
21341A4529@gmrit.edu.in

Hari Prasad Ippili
dept. of Artificial intelligence and data science(AIDS)
GMRIT
Rajam, India
hariprasadippili1234@gmail.com

Abstract**—** Human-Computer Interaction (HCI) explores the integration of voice commands and hand gestures for system control. Leveraging advancements in speech recognition, voice command technology provides an intuitive communication channel with computing devices. Simultaneously, hand gestures offer a natural, non-intrusive alternative, particularly valuable in contexts where traditional input methods are cumbersome. This paper extends the integration of voice commands and hand gestures into the Internet of Things (IoT) framework, specifically utilizing the ESP32 microcontroller. Known for its wireless communication capabilities and low power consumption, the device serves as the core of this IoT-enabled system, enabling users to control smart devices and appliances seamlessly through voice and gesture recognition. This fusion supports remote, real-time management of IoT devices, creating a more connected and intelligent environment.
Rigorous user testing, feedback analysis, and usability assessments evaluate the system's effectiveness, accuracy, and user satisfaction. The inclusion of IoT broadens the scope of HCI applications, offering practical solutions for smart homes, healthcare monitoring, and industrial automation. Additionally, the system's potential applications span diverse domains, including gaming, healthcare, education, and smart home automation, contributing to a more intuitive and accessible interaction between users and technology.

Keywords**: -** Voice command, Hand gestures, System control, Human-computer interaction (HCI), Speech recognition, Gesture-based interactions, Internet of Things (IOT).

I. Introduction

The paper explores the integration of voice command and hand gesture recognition in human-computer interaction (HCI) to offer more intuitive and natural ways of interacting with digital systems. Voice command, powered by natural language processing (NLP), enables users to control devices and execute commands using spoken instructions, making it especially useful for hands-free operation and accessibility. Hand gesture recognition uses computer vision and machine learning to interpret hand and finger movements, allowing tactile and gesture-based interaction, which complements traditional input methods.

The paper highlights the growing role of these technologies in various applications such as gaming, virtual reality, smart homes, healthcare, and tools for individuals with disabilities. Additionally, the research extends the application of HCI modalities to the Internet of Things (IoT) by using the ESP32 microcontroller—a low-power device with Wi-Fi and Bluetooth. By combining speech recognition and computer vision, the system enables real-time, hands-free control of IoT devices in smart environments. This approach offers a practical, user-friendly interface for controlling devices in smart homes, healthcare systems, and industrial automation, making HCI more accessible and effective for everyday use.

Applications of Voice Command HCI:

- Smart Homes: Voice-controlled devices like smart speakers, thermostats, and lighting systems allow users to manage their home environments effortlessly.
- Healthcare: Voice interfaces are used in healthcare for dictation of medical records, patient monitoring, and voice-controlled medical devices, improving efficiency and accessibility for healthcare professionals and patients.
- Education: Voice-controlled educational tools and language learning apps provide interactive and engaging learning experiences for students of all ages.

Applications of Hand Gesture Recognition HCI:

- Gaming and Entertainment: Gesture-based gaming consoles and VR/AR systems offer immersive gaming experiences where users can control gameplay and interact with virtual environments using natural hand movements.
- Industrial Automation: Gesture-controlled interfaces in

industrial settings improve worker safety and efficiency by enabling hands-free control of machinery, equipment, and robotic systems.

II. *Literature survey*

This collection of papers provides an overview of recent advances in hand gesture recognition, voice command systems, and their applications in human-computer interaction (HCI), particularly in conjunction with IoT devices and virtual assistants. Here's a summary of each paper:

1. Zahra, R., Shehzadi, A., Sharif, M. I., Karim, A., Azam, S., De Boer, F., Jonkman, M., & Mehmood, M. (2021). discuss a camera-based interactive wall display system that uses bare hand gestures, eliminating the need for external devices like gloves. The system consists of three modules: gesture recognition (using Genetic Algorithms and Otsu thresholding), controlling external functions, and finger counting (using the convexity hull method), focusing on efficiency and natural interaction.

2. Sánchez-Nielsen, E., Antón-Canalis, L., & Hernández-Tejera, M. (2004). aim to develop a low-cost, real-time vision system for hand gesture recognition. Their system uses skin color and blob analysis to detect hand postures, achieving 90% accuracy, though it's influenced by factors like lighting conditions and background complexity.

3.Alnilam, A., & Zakariah, M. (2022). using Reset and Mobile Net for recognizing Arabic sign language. Their system achieves high classification accuracy on the ArSL2018 dataset and is notable for its ability to handle 32 different sign language classes.

4. Badi, H. (2016). compares two feature extraction methods (hand contour and complex moments) for hand gesture recognition. While complex moments-based neural networks offer greater accuracy, hand contour-based models provide faster training speeds.

5. Xu, J., & Wang, H. (2022). Wang present a robust hand gesture recognition system based on RGB-D data. They use the Distance Transform algorithm for static gesture recognition and a combination of Euclidean distance and skeleton feature modulus ratios for dynamic gestures.

Overall, these papers collectively illustrate the progress and challenges in HCI, particularly through hand gesture recognition, voice commands, and virtual assistants, with applications in various fields including healthcare, education, and smart environments.

III. *Methodology*

Hand Gestures Recognition

1. **Data Collection**: The data is created which consists of different types of hand gestures that are created by customized ones.

2. **Hand Image:** Hand input images capture hand movements, poses, or gestures to enable natural interaction with digital devices in HCI applications. In this context, static hand input images show the hand in a specific pose or position, enhancing usability and accessibility.

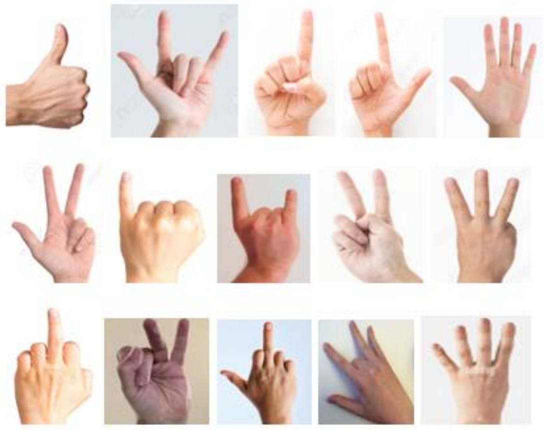


Fig 1: Hand Gesture Dataset

3. **Hand Detection:** Hand detection identifies and locates human hands in images or video frames, typically using bounding boxes or key points. It serves as the first step for further analysis, like gesture recognition or hand tracking.

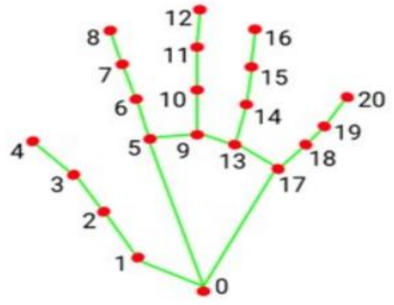


Fig 2: Hand Detection

4. **Pre-Processing:** Preprocessing in hand gesture recognition enhances input data quality before using it in a machine learning model. Key steps include

- **Image Acquisition:** Capturing hand gestures using cameras or sensors with good lighting.
- **Image Cropping:** Focusing on the hand region to remove irrelevant background.
- **Noise Reduction:** Applying techniques like Gaussian blurring to reduce image noise while preserving important features.

5. **Feature Extraction:** We used a "Hand Tracking Module" to detect, track, and analyze hand movements in applications like HCI, virtual reality, and augmented reality. The module captures video using OpenCV and performs

- **Hand Detection:** Uses techniques like color segmentation or machine learning to detect hands in images or video.

- **Hand Landmark Detection:** Identifies key points such as fingertips, knuckles, and palm points on the detected hands.
- **Finger Tracking:** Tracks finger movements by analyzing the spatial relationships between the detected landmarks.

6. **Recognition:** The system uses finger counting and the convex hull method to recognize gestures like thumbs up, pointing, or a closed fist.

- **Rule-based Classification:** Simple algorithms classify gestures based on the detected finger positions, such as recognizing a thumbs up or pointing.
- **Template Matching:** Compares hand configurations with predefined gesture templates for accurate recognition.

7. **Gesture Dictionary**: A collection of reference gestures the system recognizes, each linked to a specific command.

- **Geometric Data:** Stores locations of fingertips, palm center, and finger angles.
- Image Templates: Predefined hand images representing
gestures.
- **Feature Descriptors:** Advanced systems use key point
detection for gesture representation.
- **Command Association:** Each gesture triggers a specification, like a raised index finger for a "click" command in a virtual environment.

8. **Command:** The system executes the command associated with the recognized gesture. This may involve sending a signal to a device, performing an action on a computer, or controlling a robot.

System Commands:

- Volume Control
- Power Management
- Window Management
- Application Commands
- Other Commands

These commands are executed based on the specific hand gestures detected by the program. The code defines a mapping between finger combinations and corresponding commands. By using hand gestures as an interface, the code allows for a hands-free way to control the system and applications.

9. **Execution:** Once a gesture is recognized, the system translates the recognized gesture into a corresponding command.

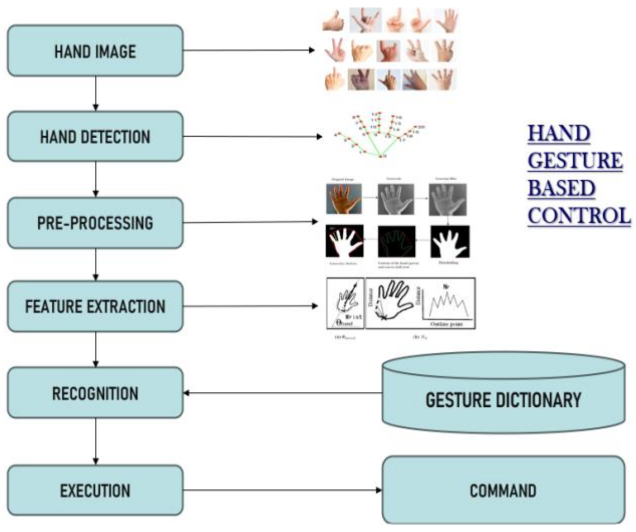


Fig 3: Hand Gesture Flow Chart

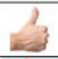
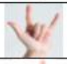
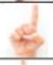
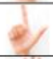
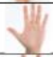
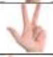
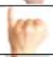
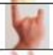
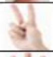
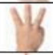
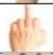
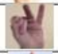
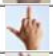
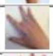
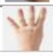
Hand Gestures	Functions
	increase_volume ()
	decrease_volume ()
	switch_window ()
	take_screenshot ()
	shut_down ()
	mute_volume ()
	unmute_volume ()
	next_slide ()
	previous_slide ()
	maximize_window ()
	minimize_window ()
	lock_screen ()
	restart_system ()
	set_timer ()
	close_window ()

Fig 4: Hand Gestures and its Functions

Voice Commands Recognition

1. **Input Voice Command:** An input voice command is a spoken instruction given to a voice control system, allowing users to convey actions verbally instead of typing. Clear and concise phrases, like "open YouTube," "increase volume," or "send a message," help the system accurately understand the desired action. This hands-free approach offers a more convenient alternative to traditional input methods.

2. **Conversion of Voice into Text using Speech Recognition Module:** After the user speaks, the voice control system employs a speech recognition module to convert audio into text. This module analyzes sound waves and matches them to patterns of words and phrases in its database.

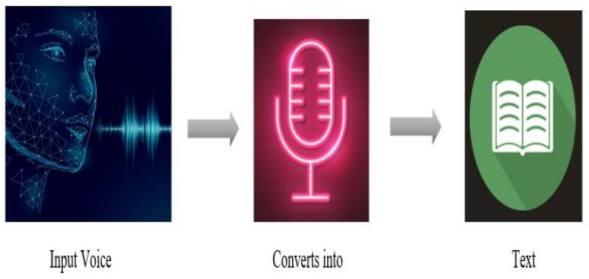
   Capturing Audio: The process starts by capturing audio input from a connected microphone.

   Preprocessing: Preprocessing steps, like reducing ambient noise, enhance the clarity of the user's voice for better recognition.

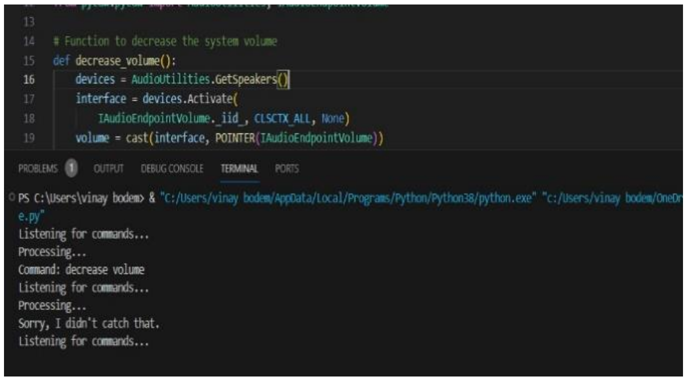
   Recognition: The preprocessed audio is analyzed by the speech recognition system, using algorithms to identify patterns of spoken words or phrases.

   Decoding: Recognized audio is decoded into phonemes or words by comparing extracted features with known speech patterns in the system's language model.

   Output: The final output is the recognized speech in text form, which can be used for executing commands, generating captions, or transcribing dialogues.



3. **Understanding the command given by the User:** Once the speech recognition module converts the voice to text, the system tries to understand the meaning of the command. This may involve tasks like identifying the keywords in the sentence and understanding the overall intent of the user.



```
# Function to decrease the system volume
def decrease_volume():
    devices = AudioUtilities.GetSpeakers()
    interface = devices.Activate(
        IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
    volume = cast(interface, POINTER(IAudioEndpointVolume))
```

```
PS C:\Users\vinay bodem> & "C:/Users/vinay bodem/AppData/Local/Programs/Python/Python38/python.exe" "c:/Users/vinay bodem/OneDr
e.py"
Listening for commands...
Processing...
Command: decrease volume
Listening for commands...
Processing...
Sorry, I didn't catch that.
Listening for commands...
```

4. **Processing the command:** After understanding the command, the system needs to process it and determine the appropriate action to take. This might involve breaking down the command into smaller steps or fetching information from external sources.

   ➢ **Command Matching and Breakdown:**
   - The system maintains a database of supported commands and their corresponding actions. When it receives a user command (like "open YouTube"), it searches this database for a match.
   - If the command is simple and well-defined (e.g., "increase volume"), the system can directly proceed to the execution stage.

   ➢ **Argument Extraction:**
   - Some commands require additional information to perform the desired action accurately. These are called arguments. For instance, opening a specific website requires the URL as an argument.

   ➢ **Function Execution:**
   - Once the system understands the command and any necessary arguments, it translates that knowledge into concrete actions. This is where pre-written functions come into play.
   - Based on the parsed command and arguments, the system triggers the appropriate function(s) to carry out the user's request.

   ➢ **System Interaction:**
   - The functions executed in the previous step interact with various components to fulfill the user's command. This interaction might involve:
   - Accessing the operating system (OS) to adjust settings (e.g., volume control) or launch applications.

5. **Checking in the Commands and Functions:** The system checks its database of commands and functions to see if it can find a match for the user's command. This database likely contains a list of all supported commands and the corresponding functions that the system should execute to perform those commands. By maintaining a well-defined command database and efficiently matching user commands with their corresponding functionalities, the system ensures it can accurately interpret user intent and execute the desired actions.

6. **Executing the command:** If the system finds a match for the user's command in its database, it executes the corresponding function. These functions are essentially a set of pre-written instructions that tell the system how to perform specific actions.

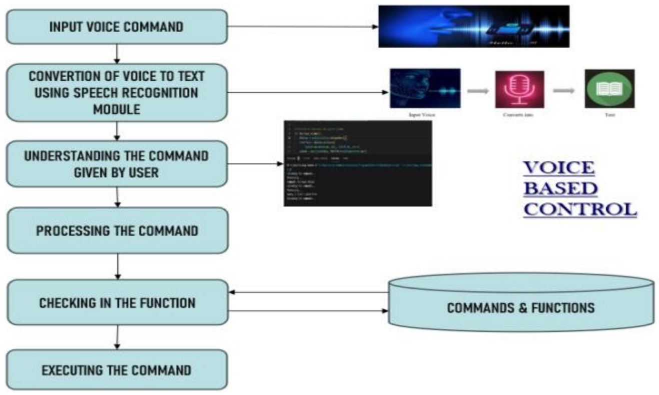


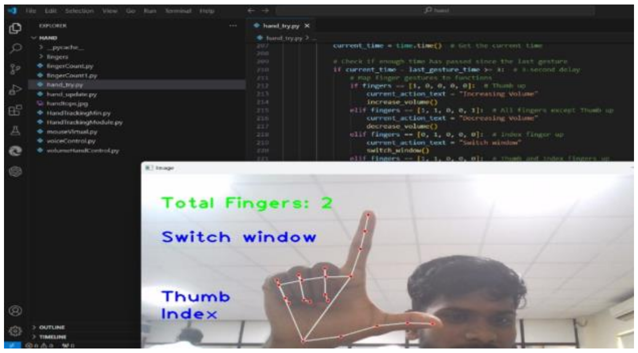
Fig 5: Voice Commands Recognition Flow Chart

Fig 5: Voice Commands Recognition Flow Chart

IV. Results

The results of the exploration into the fusion of voice commands and hand gestures for system control in human-computer interaction (HCI) reveal promising advancements in intuitive communication channels with computing devices.

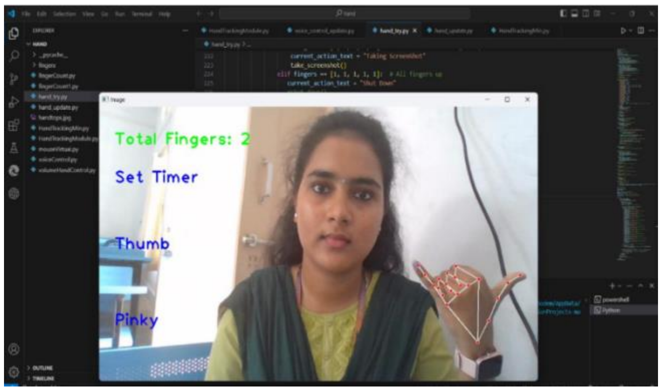
Hand Gesture Outcomes:
Users were able to interact with digital systems using natural hand movements, enabling tasks such as navigation, selection, and control of applications and devices. The system's effectiveness was evident in its ability to accurately detect and classify a variety of hand gestures, including complex movements and poses.



```
current_time = time.time()  # Get the current time

# Check if enough time has passed since the last gesture
if current_time - last_gesture_time >= 3:  # 3-second delay
    # Map finger gestures to functions
    if fingers == [1, 0, 0, 0, 0]:  # Thumb up
        current_action_text = "Increasing Volume"
        increase_volume()
    elif fingers == [1, 1, 0, 0, 1]:  # All fingers except Thumb up
        current_action_text = "Decreasing Volume"
        decrease_volume()
    elif fingers == [0, 1, 0, 0, 0]:  # Index finger up
        current_action_text = "Switch window"
        switch_window()
    elif fingers == [1, 1, 0, 0, 0]:  # Thumb and Index fingers up
```

Fig 6: Hand Gesture for Switching Window

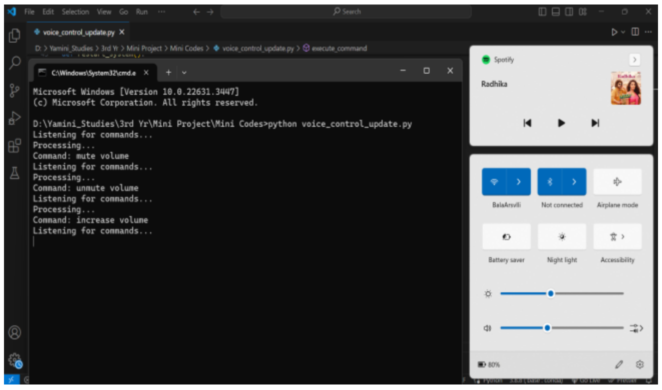


```
            current_action_text = "Taking Screenshot"
            take_screenshot()
        elif fingers == [1, 1, 1, 1, 1]:  # All fingers up
            current_action_text = "Shut Down"
```

Fig 7: Hand Gesture for Set Timer

Voice Commands Outcomes:
Users interacted effortlessly with computing devices, issuing commands for tasks like volume adjustment and application control. The system effectively interpreted a wide range of spoken instructions, even with variations in accent, tone, and speech speed.

IoT Outcomes:
Integrating IoT with voice and gesture control enabled seamless interaction with smart devices. Users effortlessly managed tasks like controlling lights and adjusting thermostats through natural gestures and voice commands, showcasing high responsiveness and reliability in real-time device management.

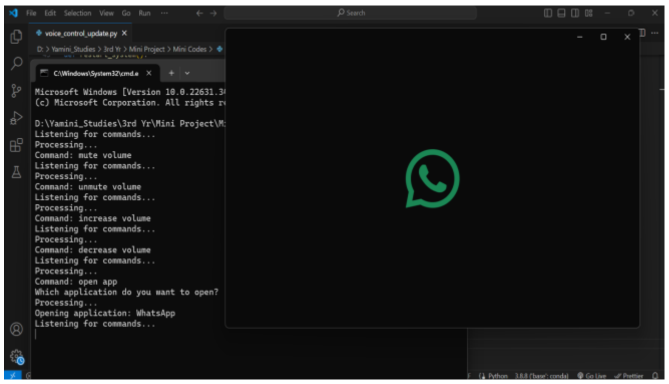


```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

D:\Yamini_Studies\3rd Yr\Mini Project\Mini Codes>python voice_control_update.py
Listening for commands...
Processing...
Command: mute volume
Listening for commands...
Processing...
Command: unmute volume
Listening for commands...
Processing...
Command: increase volume
Listening for commands...
```

voice_control_update.py
D: > Yamini_Studies > 3rd Yr > Mini Project > Mini Codes > voice_control_update.py > execute_command
C:\Windows\System32\cmd.e
Spotify
Radhika
BalaArvslli
Not connected
Airplane mode
Battery saver
Night light
Accessibility
80%

Fig 11: Voice Command For Increase Volume



```
Microsoft Windows [Version 10.0.22631.34
(c) Microsoft Corporation. All rights r

D:\Yamini_Studies\3rd Yr\Mini Project\M
Listening for commands...
Processing...
Command: mute volume
Listening for commands...
Processing...
Command: unmute volume
Listening for commands...
Processing...
Command: increase volume
Listening for commands...
Processing...
Command: decrease volume
Listening for commands...
Processing...
Command: open app
Which application do you want to open?
Processing...
Opening application: WhatsApp
Listening for commands...
```

voice_control_update.py
D: > Yamini_Studies > 3rd Yr > Mini Project > Mini Codes >
C:\Windows\System32\cmd.e

Fig 8: Voice Command for Open APP

Devices Supporting Hardware Deployment:

Rapid advancements in embedded systems and IoT technologies have facilitated the development of complex, cost-effective applications. This paper examines a multi-device hardware integration involving Arduino, ESP32 with a webcam, and the Borodino FT232RL FTDI USB to TTL Serial Converter. The combination of these components creates a robust platform for applications like wireless surveillance, smart robotics, and IoT-enabled monitoring systems, with hardware design and communication protocols enabling versatile real-time data acquisition and transmission.

1. Arduino:
Arduino is an open-source electronics platform that simplifies

electronic project development. It consists of a programmable circuit board (microcontroller) and an Integrated Development Environment (IDE) for coding and uploading programs. In this setup, Arduino controls external peripherals such as motors, sensors, and actuators.

2. ESP32 with Webcam

The **ESP32** is a low-cost, low-power microcontroller with built-in Wi-Fi and Bluetooth, making it ideal for IoT applications. The **ESP32-CAM** variant comes with an integrated camera module, which adds image and video streaming capabilities to the setup.

The ESP32-CAM, when paired with additional modules like the FTDI USB to TTL Serial Converter, allows seamless data transfer to and from a host system for debugging or further processing.

3. Roboduino FT232RL FTDI USB to TTL Serial Converter Adapter Module

The **Roboduino FT232RL FTDI USB to TTL Serial Converter** is a USB to UART interface device that facilitates serial communication between microcontrollers (like Arduino and ESP32) and a computer or other USB-enabled devices. It uses **TTL (Transistor-Transistor Logic)** for data transmission, which is ideal for microcontroller-based systems where standard USB communication is not possible.

.4. Integration and Functionality

In this system, the **Arduino** controls peripheral devices such as sensors, relays, or motors, while the **ESP32-CAM** module is responsible for capturing images and transmitting them wirelessly. The **FT232RL FTDI USB to TTL Serial Converter** enables communication between the ESP32/Arduino and a host system for programming or real-time data transmission.

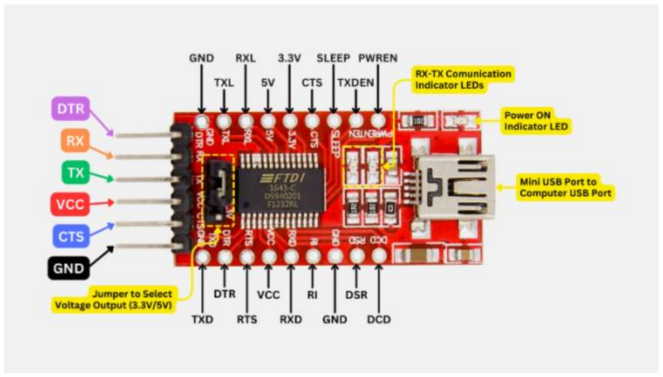

GND
RXL
3.3V
SLEEP
PWREN
TXL
5V
CTS
TXDEN
RX-TX Comunication Indicator LEDs
Power ON Indicator LED
Mini USB Port to Computer USB Port
DTR
RX
TX
VCC
CTS
GND
Jumper to Select Voltage Output (3.3V/5V)
DTR
VCC
RI
DSR
TXD
RTS
RXD
GND
DCD

Fig 9: ROBODUINO FT232RL FTDI USB to TTL

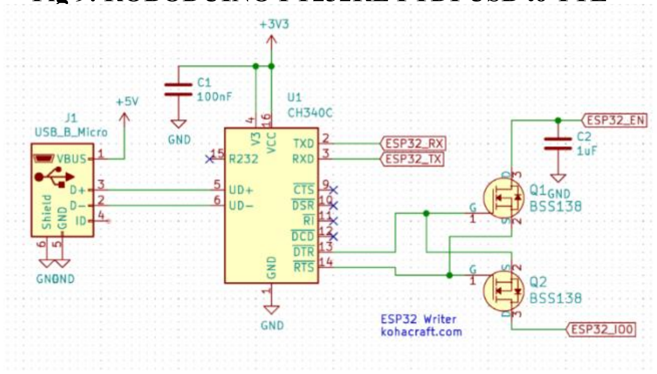


Fig 10: ESP 32 to ROBODUINO CONNECTION

Conclusion and Future Scope:

The integration of voice commands and hand gestures in human-computer interaction (HCI) offers a significant step forward in creating more intuitive communication with digital systems. Combining speech and gesture recognition allows users to easily perform tasks like navigation, selection, and system control, improving efficiency and satisfaction across domains such as gaming, healthcare, education, and smart home automation.

Despite challenges like accuracy and privacy concerns, ongoing research continues to enhance these technologies. The future of this integrated HCI approach looks promising, with potential for continued innovation and widespread adoption, providing hands-free, natural, and tactile user experiences.

References

[1] Zahra, R., Shehzadi, A., Sharif, M. I., Karim, A., Azam, S., De Boer, F., Jonkman, M., & Mehmood, M. (Year). “Camera-based interactive wall display using hand gesture recognition”.

[2] Sánchez-Nielsen, E., Antón-Canalís, L., & Hernández-Tejera, M. (2004). “Hand gesture recognition for human-machine interaction”.

[3] Siby, J. E. R. A. L. D., Kader, H. I. L. W. A., & Jose, J. I. N. S. H. A. (2015). “Hand gesture recognition. IJITR) International Journal of Innovative Technology and Research”, Volume, (3), 7-11.

[4] Panwar, M., & Mehra, P. S. (2011, November). “Hand gesture recognition for human computer interaction”. In 2011 International Conference on Image Information Processing (pp. 1-7). IEEE.

[5] Patel, Sunny, Ujjayan Dhar, Suraj Gangwani, Rohit Lad, and Pallavi Ahire. "Hand-gesture recognition for automated speech generation." In 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT).

[6] Badi, H. (2016). Recent methods in vision-based hand gesture recognition. International Journal of Data Science and Analysis.

[7] Fahad, M., Akbar, A., Fathima, S., & Bari, M. A. (2023). “Windows Based AI-Voice Assistant System using GTTS”. Mathematical Statistician and Engineering Applications.

[8] Bhargav, K. M., Bhat, A., Sen, S., Reddy, A. V. K., & Ashrith, S. D. (2022, September). Voice-Based Intelligent Virtual Assistant for Windows. In International Conference on Innovations in Computer Science and Engineering.

[9] voice-based intelligent virtual assistant for Windows usin python Rose Thomas, Surya V S, Tincy A Mathew, Tinu Thomas International Journal of Engineering Research & Technology (IJERT)

[10] Chinchane, A., Bhushan, A., Helonde, A., & Bidua, K. SARA: A Voice Assistant Using Python. International Journal for Research in Applied Science and Engineering Technology, 10(6), 3567-3582.

[11] Geetha, V., Gomathy, C. K., Vardhan, K. M. S., & Kumar, N. P. (2021). The voice-enabled personal assistant for PC using Python. International Journal of Engineering and Advanced Technology.

[12] Asodariya, H., Vachhani, K., Ghori, E., Babariya, B., & Patel, T. Desktop Voice Assist