engineering proceedings

MDPI

Proceeding Paper

Hand Gesture to Sound: A Real-Time DSP-Based Audio Modulation System for Assistive Interaction †

Laiba Khan [1,*], Hira Mariam [2], Marium Sajid [1], Aymen Khan [1] and Zehra Fatima [1]

[1] Department of Electronic Engineering, NED University of Engineering and Technology, Karachi 75270, Pakistan; sajid4604006@cloud.neduet.edu.pk (M.S.); khan4602214@cloud.neduet.edu.pk (A.K.); fatima4601094@cloud.neduet.edu.pk (Z.F.)
[2] Department of Telecommunications Engineering, NED University of Engineering and Technology, Karachi 75270, Pakistan; hiramariam@cloud.neduet.edu.pk
* Correspondence: khan4630271@cloud.neduet.edu.pk
† Presented at the 12th International Electronic Conference on Sensors and Applications, 12–14 November 2025; Available online: https://sciforum.net/event/ECSA-12.

Abstract

This paper presents the design, development, and evaluation of an embedded hardware and digital signal processing (DSP)-based real-time gesture-controlled system. The system architecture utilizes an MPU6050 inertial measurement unit (IMU), Arduino Uno microcontroller, and Python-based audio interface to recognize and classify directional hand gestures and transform them into auditory commands. Wrist tilts, i.e., left, right, forward, and backward, are recognized using a hybrid algorithm that uses thresholding, moving average filtering, and low-pass smoothing to remove sensor noise and transient errors. Hardware setup utilizes I^2C-based sensor acquisition, onboard preprocessing on Arduino, and serial communication with a host computer running a Python script to trigger audio playing using the playsound library. Four gestures are programmed for basic needs: Hydration Request, Meal Support, Restroom Support, and Emergency Alarm. Experimental evaluation, conducted over more than 50 iterations per gesture in a controlled laboratory setup, resulted in a mean recognition rate of 92%, with system latency of 120–150 milliseconds. The approach has little calibration costs, is low-cost, and offers low-latency performance comparable to more advanced camera-based or machine learning-based methods, and is therefore suitable for portable assistive devices.

Keywords: hand-gesture recognition; audio modulation; digital signal processing (DSP); assistive technology; human–computer interaction (HCI); arduino uno

check for updates

Academic Editor: Francisco Falcone

Published: 7 November 2025

Citation: Khan, L.; Mariam, H.; Sajid, M.; Khan, A.; Fatima, Z. Hand Gesture to Sound: A Real-Time DSP-Based Audio Modulation System for Assistive Interaction. *Eng. Proc.* **2025**, *118*, 27. https://doi.org/10.3390/ECSA-12-26516

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Advances in auditory computing have uncovered a range of sound synthesis methods on both mainstream and low-cost computing platforms and have thus paved the way for real-time sound creation based on computers for a wide range of applications [1]. Both signal processing and physical modeling methods have become mature enough to be used in immersive environments, interactive installations, and assistive interfaces.

Sound spatialization technology, however, has evolved from basic multi-channel configurations to sophisticated software-based systems that can emulate intricate acoustic environments [2]. Gesture-controlled spatialization provides a natural form of interaction, allowing for people to place and move sound elements using their movements. The idea has been applied to live musical performance paradigms and interactive media [3].

This effort extends this process into the assistive technology domain, creating a wearable audio system controlled by gestures to communicate primary needs through established audio messages. Employing the MPU6050 IMU, Arduino processing, and playback through Python 3.13.9, we developed a low-latency, low-cost, and simple-to-deploy system.

2. Literature Survey

2.1. Gesture Recognition Techniques Using MPU6050

The MPU6050 contains a three-axis gyroscope and a three-axis accelerometer and is thus able to sense a wide range of human motion patterns. Experiments have evidenced that filtering techniques with calibrated thresholds can achieve high recognition accuracy without having to utilize advanced machine learning models [4]. Such systems are commonly utilized in wearable electronics and assistive technology.

2.2. MPU6050 Implementation Using Arduino for Motion Detection

The sensor is connected to Arduino through I2C protocol, and libraries like Wire and MPU6050.h are used to read data [5–7]. Accelerometer and gyroscope raw values are filtered to identify tilts, rotations, or movements. Thresholds are customized by calibration to become user-specific and generalizable.

2.3. Application of MPU6050 in Audio Modulation

In audio applications, the MPU6050 becomes a hands-free control interface that can perform actions like volume control, effects application, or cue loading [8]. Such applications are beneficial to people with motor disabilities and can be integrated within interactive performance environments.

3. Challenges in Real-Time Gesture-Controlled Audio Systems

Gyroscope drift, sensor noise, and gesture overlap must be addressed while designing the system. Classifications can be in error when acceleration patterns are similar or similar in different gestures or when gestures are executed irregularly [9–11]. Latency in serial communication and audio playback also affect the user experience. User strength variability and movement accuracy also require strong calibration strategies.

4. System Overview

The four functional levels of the gesture-controlled assistive sound system are as follows:

- Sensor Layer (MPU6050): Measures acceleration and angular velocity of the wearer's wrist to sense tilts equivalent to reported specifications.
- Processing Unit (Arduino Uno): Reads sensor data, filters, and classifies gestures.
- USB Serial Communication Layer: Transmits the identified gesture label to a PC.
- Sound Output Layer (Python + Speaker): Plays an audio announcement of the requirement, for instance, "Bring water" for a Hydration Request.

5. Methodology

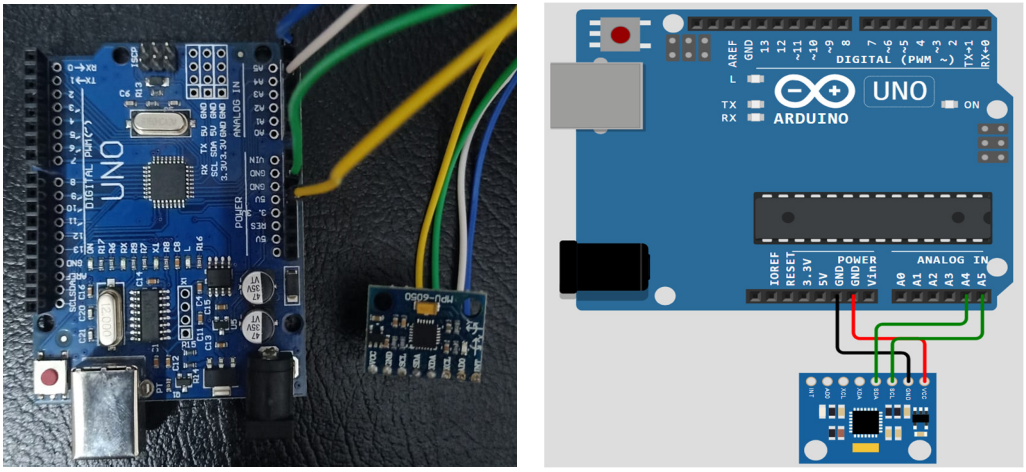
The system was deployed through synchronized hardware construction, software coding, DSP filtering, threshold tuning, and validation testing.

5.1. Hardware Setup

The MPU6050 was fixed firmly on the user's wrist via an adjustable strap. Natural wrist tilts were used as gestures in this position. The sensor was connected to the Arduino Uno with VCC connected to 5V, GND connected to GND, SDA connected to A4, and

SCL connected to A5. The Arduino was connected to the PC through USB for power and data transfer.

As provided in Figure 1, the hardware prototype (a) consists of MPU6050 and Arduino Uno with wiring connections, while the simulated wiring diagram (b), which was designed using Wokwi Simulator, shows the same connection layout.



(**a**) (**b**)

Figure 1. (**a**) Hardware prototype consisting of MPU6050 and Arduino Uno with wiring connections: VCC to 5 V, GND to GND, SDA to A4, and SCL to A5. (**b**) Wokwi simulation diagram showing the same connection layout.

5.2. System Flow

The procedure begins with ongoing data collection from gyroscopes and accelerometers. The Arduino filters data through filtering methods, identifies gestures based on pre-defined thresholds, and delivers the identified gesture label to the computer. There is a Python program that waits for the received labels, and it then plays the corresponding audio file. The complete data flow from gesture detection to sound output is depicted in the workflow diagram in Figure 2.

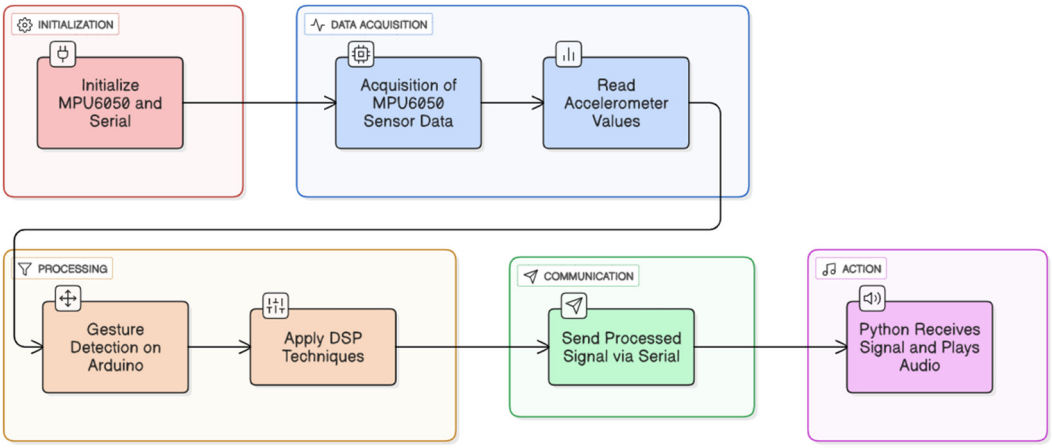


Figure 2. System flowchart illustrating the process from sensor data acquisition to audio playback.

5.3. Arduino Firmware

Filtering consisted of a low-pass filter defined by

$$y[n] = \alpha y[n-1] + (1-\alpha)x[n] \quad (1)$$

with $\alpha = 0.6$ and $0 < \alpha < 1$ to eliminate transient noise and a five-sample moving average filter

$$y[n] = \frac{1}{5}\sum_{k=0}^{4} x[n-k] \quad (2)$$

to smooth the data. Filtering uses low-pass and moving-average methods to remove transient noise and smooth sensor readings for more reliable gesture classification.

Gesture classification thresholds were specified as follows:

- Forward tilt (ay > +12,000) corresponds to a Hydration Request;
- Backward tilt (ay < −12,000) corresponds to Meal Assistance;
- Right tilt (ax > +12,000) corresponds to Restroom Assistance;
- Left tilt (ax < −12,000) corresponds to Emergency Alert.

The threshold values were determined experimentally through repeated trials to maximize recognition accuracy. The 1500 milliseconds cooldown time effectively inhibited successive activations.

5.4. Python Audio Playback and Testing Protocol

Executed using PySerial and playsound, the script watched over the COM port for gesture keywords and played back matching MP3 files. The structure provides for customization for other languages or settings.

Thresholds were determined through a sequence of iterative tests with three participants performing each gesture at varying velocities and orientations. The final test consisted of 50 repetitions of each gesture performed within a controlled laboratory environment.

6. Results and Discussion

6.1. Gesture Recognition Accuracy

The accuracy of recognition per gesture is presented in Table 1 and illustrated in Figure 3. Recognition accuracy was calculated using the following formula:

$$Accuracy\ (\%) = \frac{Number\ of\ Correct\ Classifications}{Total\ Number\ of\ Gestures\ Performed} \times 100 \quad (3)$$

where Number of Correct Classifications refers to gestures correctly recognized by the system and Total Number of Gestures Performed is the total amount of all trials per gesture. Each gesture was performed 50 times in a controlled setting, and a recognition was deemed correct if the recognized gesture was the one intended.

Table 1. Recognition rates for each gesture in laboratory-controlled tests.

Gesture	Recognition Rate (%)
Hydration Request	94.2
Meal Assistance	92.8
Restroom Assistance	91.3
Emergency Alert	89.7

In comparison, Jadhav et al. [4] reported an overall accuracy of 89.3% using an MPU6050-based glove with threshold detection, while Prasanth et al. [11] demonstrated smooth and low-latency cursor control, but did not report a numerical accuracy figure. Our mean recognition accuracy of 92.2% thus compares favorably to existing IMU-based approaches.

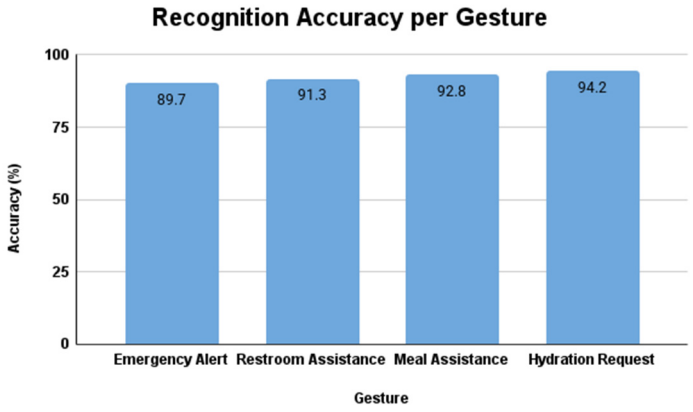


Figure 3. Recognition accuracy for each gesture in laboratory-controlled tests.

6.2. Gesture Recognition Latency

Measured latency values for all gestures are listed in Table 2. Latency was calculated as the time taken between the completion of a gesture and the beginning of audio playback from Arduino serial output and Python program logs timestamps. The latency distribution is also plotted in Figure 4.

Table 2. Average latency per gesture measured from detection to audio playback.

Gesture	Latency (ms)
Hydration Request	120
Meal Assistance	125
Restroom Assistance	140
Emergency Alert	150

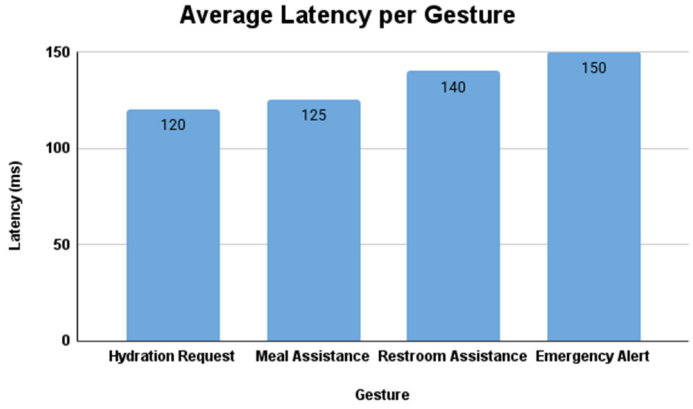


Figure 4. Average latency for each gesture measured from detection to audio playback.

6.3. Overall Performance and Interpretation

The system provided consistent real-time recognition, 92% mean accuracy, and 120–150 ms latency with all the gestures.

The maximum recognition accuracy of 94.2% was for the forward tilts (Hydration Request) and is due to a distinct acceleration profile compared to the other gestures. The lowest recognition accuracy (89.7%) was for Emergency Alert.

The lowest latency was found for Hydration Request (120 ms), perhaps because it has a clear-cut acceleration pattern and rapid detection. The highest latency was for Emergency Alert (150 ms), which is also possibly because of more sophisticated or longer wrist motion patterns prior to threshold detection. Latency tests showed that most of the delays occurred

during the audio playback stage in Python, suggesting room for improvement through other playback libraries.

In comparison to camera-based gesture recognition [4,10], this IMU-based method has the following:

- Lower hardware cost and complexity;
- Privacy-protecting operation;
- Minimize computational burdens for instant utilization.

While the system demonstrates high recognition accuracy and low latency, certain limitations remain. The gesture vocabulary is restricted to four predefined wrist tilts, which may constrain broader interaction scenarios. Misclassifications were sometimes caused by incomplete or too-rapid movements, and long-term gyroscope drift could subtly shift threshold levels, a limitation that may be alleviated through sensor fusion algorithms. Occasional recalibration may also be required when adapting the system to different users.

Furthermore, evaluation was limited to a laboratory environment with a small participant pool, and further validation in real-world assistive settings is necessary. On the other hand, the hardware remains highly cost-effective, with the MPU6050 and Arduino Uno together priced under USD 15, making the solution considerably more affordable than camera-based or EEG/EMG-based alternatives.

6.4. Comparative Analysis with the State of the Art

The current IMU-based and vision-based gesture recognition systems have a set of trade-offs in terms of accuracy, latency, cost, and portability. To put the proposed system into context, comparison with state-of-the-art methods described in the literature was performed.

Table 3 demonstrates the performance, feasibility, and limitations of the existing methods compared to the proposed system.

Table 3. Comparative analysis of the proposed system with existing state-of-the-art approaches.

System/Reference	Sensor Type and Approach	Accuracy (%)	Latency (ms)	Cost and Practicality	Notes/Limitations
Jadhav et al. [4]	MPU6050-based glove, threshold	89.3	Not reported	Moderate cost, wearable	Limited gesture set, requires glove hardware
Prasanth et al. [11]	MPU6050 for cursor movement	Not reported	Smooth, low	Low cost, simple setup	No explicit recognition accuracy reported
CNN-LSTM (Jiao et al. [10])	Vision-based (camera + ML)	>95	Higher	High cost, requires camera + GPU	Limited portability, privacy concerns
Proposed system (this work)	MPU6050 + Arduino + Python DSP	92.2	120–150 Real-time, <one-fifth of a second	Very low cost (<USD 15), wearable, portable	Limited to 4 gestures, needs real-world testing, but robust, low-latency, and cost-effective for assistive use

As shown, vision-based CNN–LSTM approaches are more accurate, but require cameras, GPUs, and powerful computation, which render them costly and less portable. IMU-based approaches, such as Jadhav et al. [4] and Prasanth et al. [11], are light, but either lack accuracy or lack recognition performance. The proposed system possesses good trade-off with 92.2% accuracy, 120–150 ms latency, and very low cost (<USD 15), hence presenting a practical and accessible technique for assistive communication.

7. Conclusions

This work suggests that a whole prototype of a gesture-based audio modulation system is implemented in assistive communication. In combination of the MPU6050 inertial measurement unit with Arduino-based preprocessing as well as audio playback using Python, the system attained an average of 92.2% recognition rate, along with associated latency that is suitable in real-time processes. Experimental analysis, with a laboratory-controlled setup and more than 50 repetitions per gesture, demonstrated balanced performance in all of the four provided commands: Hydration Request, Meal Support, Restroom Support, and Emergency Signal.

The key contributions of this work includes hardware-efficient design of a digital signal processing-based gesture recognition pipeline, a demonstration of a low-cost assistive communication system with excellent pre-defined audio prompt triggering, and a design of a calibration scheme which finds a good balance in real-world deployment between accuracy and robustness.

In practical terms, the system can be integrated into assistive devices for individuals with motor impairments, elderly patients, or those with limited speech abilities. For instance, in hospital wards or nursing homes, a wrist-mounted device could allow patients to signal hydration, meal, or restroom needs without verbal communication. Similarly, the Emergency Alert function offers a low-latency mechanism for summoning help in critical scenarios. Beyond healthcare, the same framework could extend to rehabilitation therapy, smart home environments, or hands-free control of consumer electronics. These applications emphasize the system's potential to provide cost-effective, real-time interaction where accessibility and simplicity are essential.

Applications of future work include scaling the gesture vocabulary to accommodate higher numbers of command types, using adaptive learning to adjust thresholds of recognition per user, multimodal output to raise usability further, and a wearable, battery-operated variant suitable for longer-term deployments beyond small experiments.

Author Contributions: Conceptualization, L.K. and H.M.; methodology, L.K.; software, L.K., M.S. and A.K.; validation, L.K., M.S. and Z.F.; formal analysis, L.K.; investigation, L.K., M.S. and A.K.; resources, H.M.; data curation, L.K. and Z.F.; writing—original draft preparation, L.K.; writing—review and editing, M.S., Z.F. and H.M.; visualization, L.K.; supervision, H.M.; project administration, L.K. and H.M.; funding acquisition, not applicable. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors thank the Department of Electronic Engineering and the Department of Telecommunications Engineering at NED University of Engineering and Technology for providing laboratory facilities, guidance, and technical support throughout the project.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Marshall, M.T.; Malloch, J.; Wanderley, M. Gesture Control of Sound Spatialization for Live Musical Performance. In *Gesture-Based Human-Computer Interaction and Simulation, Proceedings of the 7th International Gesture Workshop, Lisbon, Portugal, 23–25 May 2007*; Revised Selected Papers; Lecture Notes in Computer Science; Springer Nature: Berlin, Germany, 2009; Volume 5085, pp. 227–238.
2. Valbom, L.; Marcos, A. Wave: Sound and Music in an Immersive Environment. *Comput. Graph* **2005**, *29*, 871–881. [CrossRef]

3. Preview of: Audio Culture: Readings in Modern Music. Available online: https://api.pageplace.de/preview/DT0400.9781134379705_A23775385/preview-9781134379705_A23775385.pdf (accessed on 9 August 2025).
4. Jadhav, P.R.; Sable, P.N. Hand Gesture Recognizer Smart Glove Using ESP32 and MPU6050. *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)* **2024**, *12*, 641–645.
5. InvenSense Inc. MPU-6000 and MPU-6050 Product Specification, Revision 3.4, August 2013. Available online: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf (accessed on 9 August 2025).
6. Arduino. Wire—Two Wire Interface. Available online: https://docs.arduino.cc/language-reference/en/functions/communication/wire/ (accessed on 9 August 2025).
7. Random Nerd Tutorials. Arduino Guide for MPU-6050 Accelerometer and Gyroscope Sensor. Available online: https://randomnerdtutorials.com/arduino-mpu-6050-accelerometer-gyroscope/ (accessed on 9 August 2025).
8. MPU6050 Motion Sensor. SlideShare. Available online: https://www.slideshare.net/slideshow/mpu6050motionsensorpptx/258019442 (accessed on 9 August 2025).
9. Lishan, C.L.; Saravana, M.K.; Rao, S.P.; Manushree, K.B. A Comprehensive Survey on Gesture-Controlled Interfaces: Technologies, Applications, and Challenges. *Int. J. Sci. Res. Sci. Technol.* **2025**, *12*, 1112–1136. [CrossRef]
10. Hakim, N.L.; Shih, T.K.; Kasthuri Arachchi, S.P.; Aditya, W.; Chen, Y.-C.; Lin, C.-Y. Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model. *Sensors* **2019**, *19*, 5429. [CrossRef] [PubMed]
11. Raja, S.; Sinha, R.; Sharma, A.; Shrivastava, K.; Prasanth, N.N. Gesture-Based Mouse Control System Based on MPU6050 and Kalman Filter Technique. *Int. J. Intell. Syst. Technol. Appl.* **2023**, *21*, 56–71. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.