Integrating AI and IoT in STEM Education Through a Gesture-Control Project

Serhii Petrovych, Stefan-Daniel Horvath and Chunfang Zhou
STEM Education Research Center, Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, 5230 Odense, Denmark
petrovych@imada.sdu.dk, chzh@sdu.dk, horvathstefandaniel@gmail.com

Keywords: STEM Education, Hand Gesture Control, Computer Vision, Internet of Things.

Abstract: This paper presents the development and deployment of a hand gesture-controlled lighting system specifically designed for STEM education at the undergraduate level. The primary objective of this study is to demonstrate a practical framework for integrating complex AI and IoT concepts through a hands-on, constructionist learning approach. The project combines affordable microcontroller hardware, specifically the ESP32-CAM and ESP32 Dev Board, with the MediaPipe framework and Internet of Things (IoT) protocols to transform human hand movements into interactive visual effects. By utilizing MediaPipe for real-time hand landmark detection and implementing explicit geometric rules for gesture classification, the project provides students with direct experience in hardware-software integration and distributed system control. The study details the technical architecture, including robust debouncing mechanisms to ensure operational stability. Furthermore, the system supports versatile deployment options, such as standalone executable files for Windows, enhancing classroom accessibility. This "white-box" design principle facilitates a deeper understanding of embedded programming and practical AI applications. Ultimately, the project serves as a comprehensive educational tool that successfully bridges theoretical knowledge and applied STEM skills in modern engineering curricula.

1 INTRODUCTION

The rapid expansion of artificial intelligence (AI) and the Internet of Things (IoT) is reshaping STEM education by positioning learners as designers of cyber-physical systems [1], [2], with project-based approaches further promoting deeper conceptual understanding [3], [4].

Recent work has explored gesture recognition in education. Umadevi et al. [5] presented GestureMath AI, which uses MediaPipe to enable students to draw equations in the air, demonstrating the growing interest in touchless educational interaction. A 2024 study by Chen et al. [6] proposed a continuous recognition algorithm using MediaPipe BlazePose and simple rules to detect teachers' hand signals for students with attention deficits, achieving 88.3% accuracy. Similarly, Bastos et al. [7] developed a low-cost social robot using MediaPipe and OpenCV to recognize and mimic gestures for STEM learning environments.

On the algorithmic side, researchers have compared rule-based and ML approaches for gesture recognition. Malik et al. [8] presented a rule-based system for recognizing static Polish Sign Language gestures in Extended Reality (XR) using hand tracking. Arboleda et al. [9] explored opening the ML black box for multidisciplinary students through scaffolding from GUI to coding, emphasizing pedagogical approaches to making AI transparent. Wang and Chen [10] employed explainable AI methods to unpack deep learning-based classroom discourse models, enhancing STEM teachers' trust and technology acceptance.

However, most existing works rely on black-box neural networks trained on large datasets or require expensive hardware (HMDs, VR controllers). None provide a fully transparent, low-cost pipeline from ESP32-CAM to IoT actuation that students can modify at the rule level without retraining models.

This study operationalizes this integration through a gesture-controlled lighting system based on low-cost microcontrollers and computer vision. Its architecture consists of three components: an ESP32-CAM streaming video via WiFi, a computer processing the stream with Python, and an ESP32 controlling an LED strip. This distributed system makes the information trajectory explicit - from gesture to command - supporting computational thinking [2], [11].

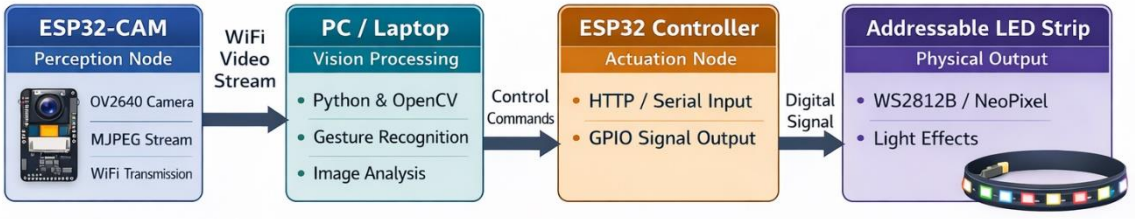

Figure 1: Block diagram of the hardware architecture for the gesture-controlled lighting system.

However, even in such systems, students often miss the internal logic due to the 'black-box' effect: most AI tools hide data processing and hardware interaction. Existing STEM toolkits treat AI as an opaque module, limiting students' understanding of the full sensing-to-actuation pipeline. This study offers a transparent 'white-box' architecture where mechanisms can be examined and modified. Students engage with image preprocessing, classical face detection [12], and AI-based hand tracking using MediaPipe [13]. Gesture classification uses geometric relations between hand landmarks, linking mathematical modeling with embodied interaction [14].

Thus, the research problem is: how to design a low-cost gesture-controlled IoT system that makes AI decision-making and hardware interaction fully transparent for undergraduate learners? The aim is to empirically investigate how a transparent, rule-based gesture recognition architecture influences students' understanding of AI-IoT integration.

Research questions:

1) Can students explain the relationship between hand landmarks, geometric rules, and LED actuation?
2) What are the main implementation challenges (lighting, debouncing) in real classroom conditions?
3) To what extent can students modify the system without prior ML knowledge?

Specific research objectives:

1) To design and implement a transparent, rule-based hand gesture recognition system using MediaPipe and ESP32, and evaluate its classification accuracy under varying lighting conditions.
2) To assess, during Microplayer Club sessions at the University of Southern Denmark, how the white-box architecture enables learners to explain the sensing-to-actuation pipeline and modify the system without prior ML expertise.
3) To identify and document the main technical challenges (debouncing stability, Wi-Fi latency, lighting sensitivity) for classroom deployment.

2 HARDWARE ARCHITECTURE

The hardware foundation of the gesture-control system is structured as a modular distributed IoT configuration that separates perception and actuation while maintaining wireless coordination. This architectural choice reflects fundamental IoT principles, including layered organization, device autonomy, and network-based interoperability. At the same time, it embodies the “sense-process-act” paradigm characteristic of cyber-physical systems, where embedded devices interact dynamically with physical processes through computational mediation.

To clarify the interaction between modules, Figure 1 illustrates the distributed hardware topology of the system.

The system is implemented using accessible and widely adopted components: an ESP32-CAM module (AI Thinker model with OV2640 camera) functioning as the perception node; a standard ESP32 development board operating as the actuation controller; an addressable LED strip such as WS2812B (NeoPixel) or a simpler digital strip; and a regulated 5V power supply, preferably external when extended LED strips are used. The ESP32 platform integrates WiFi connectivity and sufficient processing capacity for IoT prototyping, making it particularly suitable for educational experimentation with edge devices.

The ESP32-CAM captures visual data and streams MJPEG video over HTTP within a local WiFi network. This configuration allows real-time frame acquisition by Python-based computer vision scripts running on a computer. The integration of camera, antenna, and microcontroller on a single board reduces assembly complexity while preserving architectural transparency. From a systems perspective, the module operates as an autonomous sensing unit within a distributed network.

The second ESP32 board functions as the actuation node. It receives control commands via HTTP requests or serial communication and regulates lighting behavior accordingly. When paired with WS2812B LEDs, the controller generates precisely timed digital signals to address individual RGB pixels. This

demonstrates serialized data transmission and time-dependent signal encoding, central concepts in embedded and physical computing environments. Firmware parameters such as the number of LEDs and strip type allow the system to be reconfigured without altering the hardware structure, reinforcing abstraction principles in embedded design.

3 SOFTWARE ENVIRONMENT DEPENDENCIES AND GESTURE RECOGNITION

3.1 Software Architecture

The gesture-controlled IoT system follows the classical "sensing - processing - actuation" model widely used in embedded and cyber-physical systems. The software environment requires Python 3.10+ with OpenCV, MediaPipe, and requests libraries. Project dependencies are managed through a virtual environment. Figure 2 shows the overall connection diagram.

The STEM gesture-controlled IoT system requires a carefully prepared software environment. Python 3.9, 3.10, or 3.11 (64-bit recommended) is required because the gesture recognition scripts, hand_gesture.py and gesture_launcher.py, as well as MediaPipe 0.10+, depend on these versions [5].

3.2 Arduino/ESP32 Firmware and Project Configuration

The hardware component consists of two ESP32 modules. One module is the ESP32-CAM, which captures video frames and streams them via MJPEG over WiFi. This module is programmed using Arduino IDE and requires libraries for web serving and WiFi connectivity. The other module controls the LED strip. It receives HTTP commands or serial inputs from the PC to turn LEDs on or off, adjust brightness, or run fun effects. For addressable LED strips such as WS2812B, the Adafruit NeoPixel library is required. Both sketches rely on a secrets.h file that contains WiFi credentials, allowing easy configuration of network settings without modifying the main code.

Students gain experience with distributed system principles, as one device acts as the “eye” capturing visual data, while the other functions as the “hand” performing actions. Configuration parameters, such as stream URLs and LED controller URLs, are stored in a local configuration file used by the gesture launcher to persist settings across sessions.

3.3 Hand Gesture Recognition and Data Flow

Hand gesture recognition is implemented in the script hand_gesture.py using MediaPipe Hand Landmarker. This pre-trained model detects 21 three-dimensional landmarks per hand, including the wrist, four points of the thumb, and four points for each of the other fingers. The system does not require additional training. Finger states are computed geometrically to determine whether each finger is extended or folded. The thumb state is evaluated based on handedness, while the other fingers are determined by comparing the tip position to the PIP joint, allowing gestures to be recognized even when the hand is turned sideways.

Gesture classification is rule-based. For example, a pinch is detected when the distance between the thumb and index finger is less than 30% of the hand size while the other fingers are folded. An “OK” gesture uses the same pinch criteria but with other fingers extended. A fist corresponds to all fingers folded, while an open hand is recognized when four or more fingers are extended. Other gestures such as thumb up or down, pointer, peace, and rock are similarly defined. To prevent flickering, gestures are sent to the LED strip only after they remain consistent across several consecutive frames.

The data flow is as follows. The ESP32-CAM captures video and sends frames to the PC running hand_gesture.py. MediaPipe extracts hand landmarks and computes finger states. The gesture mapping module translates finger states into explicit LED commands, which are sent to the ESP32 LED controller. The controller updates the LED strip according to the commands. This design demonstrates the integration of AI-assisted detection, explicit rule-based logic, and hardware actuation within a STEM educational context.

3.4 Development and Participants

The system was developed by the authors of this study at the STEM Education Research Center - FNUG (Framing New Understanding for Growth), Department of Mathematics and Computer Science, University of Southern Denmark (SDU). A total of 12 undergraduate students from various departments at SDU participated voluntarily in the pilot study. Five sessions were conducted, each lasting 1.5 to 2 hours. Students worked in pairs following a constructionist learning protocol. No prior knowledge of computer vision, IoT, or machine learning was required. Data were collected through direct observation, system logs, and a short exit survey.

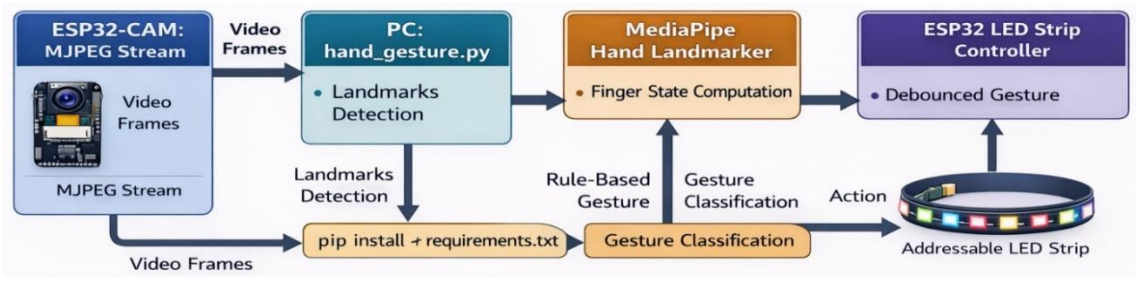

Figure 2: Block diagram of the hand gesture-controlled IoT system.

4 INTERACTIVE CONTROL AND GESTURE MAPPING

4.1 Gesture Recognition Logic

The interactive dimension of this STEM project emerges from transforming human hand movements into actions on the LED lighting system. The recognition process begins with the MediaPipe Hand Landmarker, which detects twenty-one three-dimensional landmarks on each hand, including the wrist, thumb, and four points on each of the other fingers. Gesture classification does not rely on a black-box neural network. Instead, explicit geometric rules determine the state of each finger, with the thumb using handedness to distinguish extension, and the other fingers evaluated relative to their PIP joints or distance from the wrist. Gestures are defined as specific combinations of extended and folded fingers. To ensure stability and prevent flickering, gestures are sent to the LED controller only after they are consistently observed across five consecutive frames (debouncing).

4.2 Gesture-to-Action Mapping and Creative Effects

Open Hand. An open palm triggers the ON command, illuminating the LED strip in solid color mode (see Fig. 3). This gesture represents a direct mapping from a fully extended hand to a simple visual output.

OK Gesture. The OK gesture, detected by the pinch distance between thumb and index while other fingers are extended, sends the OFF command, turning off all lighting and clearing active visual effects (see Fig. 4).

Thumb Up / Thumb Down. These gestures adjust brightness dynamically (see Fig. 5). The vertical orientation of the thumb tip relative to its joint distinguishes between increasing (Thumb Up) and decreasing (Thumb Down) light intensity.

Figure 3: Example of the Open Hand gesture controlling the LED strip.

Figure 4: Example of the OK gesture turning off the LED strip.

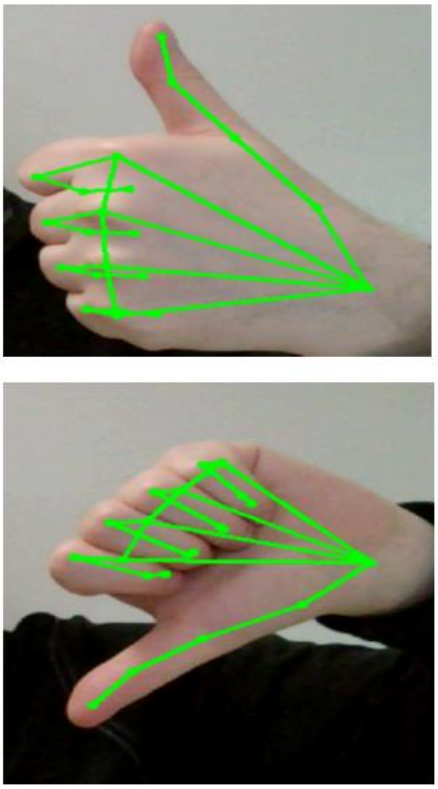

Figure 5: Thumb Up/Down gestures controlling LED brightness.

Pointer. The pointer gesture activates Fun Mode (see Fig. 6). Horizontal movement shifts the light cluster along the strip, while vertical movement changes its hue. This enables dynamic and interactive light positioning.

Figure 6: Pointer gesture controlling the position and hue of the LED cluster.

Peace Sign. The peace gesture, with index and middle fingers extended, produces a pulsating wave of light at the hand's location (see Fig. 7). The speed and duration of the wave correspond to the speed of hand movement.

Figure 7: Peace gesture producing a pulsating light wave.

Three Sign. The three-finger gesture creates a breathing effect (see Fig. 8). Light intensity gradually increases and decreases, producing a rhythmic visual pattern.

Figure 8: Three Sign gesture for breathing effect.

Rock Sign. The rock gesture triggers a strobing effect, with rapid flashes of light along the LED strip (see Fig. 9).

Figure 9: Rock gesture activating strobe mode.

Hang Loose. The hang loose gesture activates a rainbow effect, smoothly cycling through all colors of the spectrum along the LED strip (see Fig. 10).

Figure 10: Hang Loose gesture for rainbow effect.

This design allows learners to connect physical gestures with immediate visual feedback, explore dynamic interaction modes, and understand the principles of real-time control, debouncing, and human-computer interaction. The combination of rule-based gesture classification and creative lighting effects demonstrates the pedagogical value of transforming abstract hand movements into tangible, observable actions in STEM projects.

5 EXPERIMENTAL RESULTS AND CORE FINDINGS

We evaluated gesture recognition accuracy under two lighting conditions (well-lit: >300 lux, low-light: <100 lux) using 100 trials per gesture performed by three students. Table 1 summarizes the results.

Table 1: Gesture recognition accuracy (%).

Gesture	Well-lit	Low-light
Open Hand	96	81
OK	94	78
Thumb Up/Down	91	74
Pointer	89	70

The debouncing mechanism (requiring 5 consecutive consistent frames) eliminated 92% of flickering errors. During the five Microplayer Club sessions, 10 out of 12 students successfully explained how geometric rules map hand landmarks to LED commands. Eight students added a new custom gesture (e.g., "victory" to toggle rainbow mode) within 20 minutes, demonstrating the white-box modifiability. The main reported difficulties were sensitivity to background lighting and Wi-Fi latency (average 120 ms).

Core findings:

1) Rule-based gesture recognition is sufficiently transparent for students to modify and extend without ML training.
2) Lighting robustness remains the main limitation (accuracy drops 10-15% in low light).
3) Debouncing is critical for real-world stability and should be explicitly taught in STEM curricula.

6 CONCLUSIONS

This study showed that a gesture control system where students can see and change the internal logic (instead of using a black-box AI) works well for teaching AI and IoT. In sessions with 12 students from different departments at the University of Southern Denmark, 10 participants could explain how hand landmarks turn into LED commands, and 8 added a new gesture within 20 minutes without any prior machine learning training.

Accuracy was above 90% in good light (>300 lux) but dropped to 70-81% in low light (<100 lux). The debouncing mechanism (requiring 5 stable frames) removed 92% of flickering errors and proved essential for classroom use. Wi-Fi latency (average 120 ms) was another challenge, but it did not stop successful interaction.

These results have three implications for STEM education. First, keeping the decision logic transparent

helps students understand and modify the system. Second, topics like debouncing and environmental robustness should be taught explicitly in IoT courses. Third, the low-cost hardware (ESP32-CAM, ESP32, LED strip) makes the project accessible even with small budgets.

Limitations: small sample (N=12), short sessions (five 1.5-2 hour meetings), and the simple rule-based approach does not work for complex dynamic gestures. Wi-Fi dependency may be a problem in crowded classrooms. No control group using a black-box AI system was included.

Future work will improve lighting robustness (e.g., automatic gain control) and reduce Wi-Fi latency by moving gesture recognition to the ESP32-CAM edge device. A larger study with more students is also planned.

In conclusion, this gesture-controlled lighting system provides a practical, low-cost way to integrate AI and IoT into undergraduate STEM education. It bridges theory and hands-on practice while empowering students to explore, understand, and modify intelligent systems.

REFERENCES

[1] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, 1980.

[2] J. M. Wing, "Computational thinking," Communications of the ACM, vol. 49, no. 3, pp. 33-35, 2006, [Online]. Available: https://doi.org/10.1145/1118178.1118215.

[3] P. C. Blumenfeld, E. Soloway, R. W. Marx, J. S. Krajcik, M. Guzdial, and A. Palincsar, "Motivating project-based learning: Sustaining the doing, supporting the learning," Educational Psychologist, vol. 26, no. 3-4, pp. 369-398, 1991, [Online]. Available: https://doi.org/10.1080/00461520.1991.9653139.

[4] C. E. Hmelo-Silver, "Problem-based learning: What and how do students learn?" Educational Psychology Review, vol. 16, pp. 235-266, 2004, [Online]. Available: https://doi.org/10.1023/B:EDPR.0000034022.16470.f3.

[5] M. Umadevi, P. Nikitha, T. S. Goud, M. Abhinay, and L. Swathi, "Gesture Math AI: Real-time math problem solving using hand gestures and computer vision," International Research Journal on Advanced Engineering Hub, vol. 3, no. 5, 2025, [Online]. Available: https://doi.org/10.47392/IRJAEH.2025.0320.

[6] I. D. S. Chen, C.-M. Yang, S.-S. Wu, C.-K. Yang, M.-J. Chen, C.-H. Yeh, and Y.-H. Lin, "Continuous recognition of teachers' hand signals for students with attention deficits," Algorithms, vol. 17, no. 7, p. 300, 2024, [Online]. Available: https://doi.org/10.3390/a17070300.

[7] A. C. D. O. Bastos, M. F. Pinto, R. C. Coutinho, F. L. Silva, A. A. D. Lima, and G. M. Araujo, "A low-cost social robot for gesture-based educational activities and human-robot interaction in learning environments," in Proc. XVII Simpósio Brasileiro de Robótica e Simpósio Latino Americano de Robótica (SBR/LARS), pp. 408-413, 2025, [Online]. Available: https://doi.org/10.1109/sbr/wre66973.2025.11249546

[8] W. Malik, G. Kulis, and K. Skabek, "Recognition of Polish sign gesture language in extended reality," in Proc. European Conference on Modelling and Simulation (ECMS), pp. 645-652, 2025, [Online]. Available: https://doi.org/10.7148/2025-0645.

[9] M. Arboleda, C. Vieira, and J. L. Chiu, "Opening the machine learning black box for multidisciplinary students: Scaffolding from GUI to coding," in Proc. IEEE Frontiers in Education Conference (FIE), pp. 1-5, 2023, [Online]. Available: https://doi.org/10.1109/FIE58773.2023.10343043.

[10] D. Wang and G. Chen, "Making AI accessible for STEM teachers: Using explainable AI for unpacking classroom discourse analysis," IEEE Transactions on Education, vol. 67, no. 6, pp. 907-918, 2024, [Online]. Available: https://doi.org/10.1109/TE.2024.3421606.

[11] J. M. Wing, "Computational thinking and thinking about computing," Philosophical Transactions of the Royal Society A, vol. 366, no. 1881, pp. 3717-3725, 2008, [Online]. Available: https://doi.org/10.1098/rsta.2008.0118.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. I-511, 2005, [Online]. Available: https://doi.org/10.1109/cvpr.2001.990517.

[13] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays et al., "MediaPipe: A framework for building perception pipelines," arXiv preprint arXiv:1906.08172, 2019, [Online]. Available: https://doi.org/10.48550/arxiv.1906.08172.

[14] D. Long and B. Magerko, "What is AI literacy? Competencies and design considerations," in Proc. CHI Conference on Human Factors in Computing Systems, pp. 1-16, 2020, [Online]. Available: https://doi.org/10.1145/3313831.3376727.