iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VI **Month of publication:** June 2026

DOI: https://doi.org/10.22214/ijraset.2026.83923

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VI June 2026- Available at www.ijraset.com

Design and Development of Multifactor Biometric Verification for Transparent Elections

Janardhana S Y[1], Thanusha J[2], Thejaswini M U[3], Varshini S[4], Yashwanth P M[5]
Department of ECE, PESCE, Mandya, India

Abstract: ***We built a standalone verification terminal to fix a massive flaw found at standard polling places: manual ID checking. Relying on hand-checked documents makes it incredibly easy for voter impersonation, clerical mistakes, and double-voting to compromise an election. To secure this process without slowing down lines or causing a frustrating user experience, our design enforces a two-stage biometric checkpoint right at the gate. The physical prototype is straightforward. We wired an ESP32 microcontroller directly to an R307 optical fingerprint scanner and a standard digital camera module. On the software side, a local database handles data logging on site, indexing registered voter profiles and tracking live ballot updates. When someone steps up to vote, the terminal first triggers a local facial recognition script to pull up a primary database match. The system keeps access locked down until the voter backs this up with an immediate physical fingerprint scan. While the biometric checks process, the backend runs instant database queries to catch and block duplicate ballot attempts on the spot. Because we engineered this platform strictly as an outer authentication layer, it integrates with existing election equipment without requiring any core infrastructure changes. Built entirely from affordable, off-the-shelf components and open-source code, our prototype offers a highly secure, budget-friendly validation tool tailored for university and institutional voting.***

Keywords: Biometric Authentication, Electronic Voting, Face Recognition, Fingerprint Verification, ESP32, Election Security, Multi-Factor Authentication

I. INTRODUCTION

Voting is how we make decisions in groups, like in countries, companies, or even just our own college elections. The fundamental problem with all these systems is the same one: how do you prove that the person that is voting is who they say they are? Right now, most places still rely on paper lists or ID cards. It's a bit outdated and, honestly, it's really easy for someone to mess up, lose their card, or even cheat by voting twice.

That's why we're seeing a shift toward biometrics. Instead of just trusting a piece of plastic, we can use stuff like fingerprints or facial features since they're unique to every person. It's way more accurate than the old way. But, if you just use one sensor, you run into problems—like if the camera is in a dark spot, or the finger scanner isn't working right. We thought it would be much safer to just combine both methods.

For our mini project we came up with a two stage system that uses both face recognition and fingerprint scan to verify voters. The goal was to check everyone properly before they even see the voting screen. In our setup, it works like this: first, the system scans the face with a normal laptop camera using Python and OpenCV. If that passes, it moves to the next step, where an ESP32 microcontroller checks the fingerprint. By linking these two, we can stop a lot of the fake voting that happens in manual systems. We also made sure to keep the hardware simple and cheap, so it's something that could actually be used in smaller places like a college campus.

II. LITERATURE SURVEY

We looked at how others have dealt with voting security. It's pretty obvious that everyone is trying to solve the same two problems: fake votes and people trying to vote twice. Since manual check-ins are so outdated, most people are moving toward biometrics like fingerprints or face scans. They're just way more reliable since they're unique to each person and harder to fake. In [1], they built a system using only a fingerprint scanner. It was a good step up from old-school methods, but we noticed a major weak point: if the scanner didn't get a perfect read, the whole system was basically useless. In [2], a project combined face recognition with fingerprint scanning. This really clicked for us. Using two different ways to check someone is a much better safety net, and it showed us that adding that second layer of security makes the whole process way more reliable. Another project [3] looked into online voting. Their main focus was on making things easier for the user, but it was a good reminder that if your identity check isn't rock-solid, the whole fairness of the election is at risk. In [4], another fingerprint-based system was made to stop people from voting twice.

It just checked the fingerprint before someone could cast their vote, which was a pretty simple but effective way to keep things honest. After going through all of this, we realized that while biometric methods are much better than manual ones, relying on just one sensor is a huge gamble. That's why we decided to use both face recognition and a fingerprint scan for our project—we wanted a two-step check that actually makes sure the person standing there is who they claim to be before we let them vote.

III. PROBLEM STATEMENT AND OBJECTIVE

A. *Problem Statement*

The way most places handle voting is still super outdated. You've got people relying on physical ID cards or someone just looking at a list and manually checking names. It's not just slow—it's full of holes. It's way too easy for someone to make a mistake, or even worse, for someone to fake their identity or vote twice. It just makes the whole process feel less secure than it should be.

We wanted to fix that. We figured we needed a way to actually prove who is standing in front of the machine, so we built a system that uses both a face scan and a fingerprint. It's a pretty simple idea: the system forces you to pass both checks before it lets you vote. It's way harder to cheat this way than with just a plastic card. We also made sure to use cheap, simple hardware, so it's actually realistic to use this for stuff like local or college elections.

B. *Objective*

1) To design a low-cost multi-factor biometric voter verification system using an ESP32-based fingerprint module and a laptop camera for face recognition
2) To design a low-cost multi-factor biometric voter verification system using an ESP32-based fingerprint module and a laptop camera for face recognition.
3) To develop software modules for voter registration, biometric data capture, and secure storage of fingerprint and facial features in a database with basic duplicate-detection checks.
4) To implement a two-step authentication workflow at the time of voting, where both fingerprint and face must match the stored records before a vote is allowed.
5) To integrate the embedded hardware (ESP32 and fingerprint sensor) with a Python/Flask application over serial or network communication for real-time verification.
6) To evaluate the system on a small set of test users and demonstrate that it can reduce impersonation and duplicate voting compared to manual ID checking in an institutional setting.

IV. METHODOLOGY

We developed this system to address security gaps in standard voting procedures by requiring mandatory identity verification prior to ballot access. Instead of just flashing an ID card or having someone check a list, our system forces you to pass a face scan and a fingerprint check first. The whole thing runs in three parts: registration, the actual verification, and then the voting itself.

A. *Getting Registered*

Everything starts here. We start by registering the voter, taking their details and saving them to create a profile for later use. This involves capturing a facial image via a laptop camera and collecting biometric data through an ESP32-linked fingerprint sensor, which is then stored in the system for future authentication. We store all that in the system so it's ready to pull up on election day. Since the robot can move around, it is not limited to one place and can check different spots easily. Because of this, it is more helpful than fixed systems that cover only a small area.

B. *Checking the Voter*

When it's time to vote, the system acts as a gatekeeper. First, it uses Python and OpenCV with the webcam to see if the person's face matches what we have on file. If the face scan goes through, the system then asks for a fingerprint. You've got to clear both steps—the face scan and the fingerprint—before the system unlocks the voting screen.

C. *Casting the Vote*

Once the system is happy with both checks, you're cleared to vote. As soon as you finish, the system flags you as "voted" so you can't jump back in and try to vote again. If the camera doesn't recognize your face or the sensor doesn't match your print, the system just stops you right there—no access, no vote.

D. Equation

In this project, the voter is allowed to continue for voting only after both face recognition and fingerprint verification are completed successfully. Since the system checks the voter in two steps, the authentication process can be represented as:

$$V = F \backslash Times\ P$$

- (V) represents the final voter authentication result
- (F) represents the face recognition result
  ((F = 1) if the face matches and (F = 0) if it does not match)
- (P) represents the fingerprint verification result
  ((P = 1) if the fingerprint matches and (P = 0) if it does not match)

The voter is allowed to move to the voting process only when:

$$V = 1$$

This simply means that both face recognition and fingerprint verification should match successfully before the voter is allowed to vote.

V. BLOCK DIAGRAM

Traditional voting systems that rely on a single check—whether it's just scanning a physical ID card or taking one biometric reading—possess an obvious security vulnerability. If an attacker manages to forge that single credential, the system fails. To eliminate this issue entirely, our design enforces a sequential verification pipeline that chains facial recognition and physical fingerprint scanning into a mandatory, back-to-back loop. This architectural design ensures that compromising a single biometric trait is not enough to game the system. The practical execution of this framework is split into three main operational phases: voter onboarding, backend feature extraction, and live identity verification at the polling station.

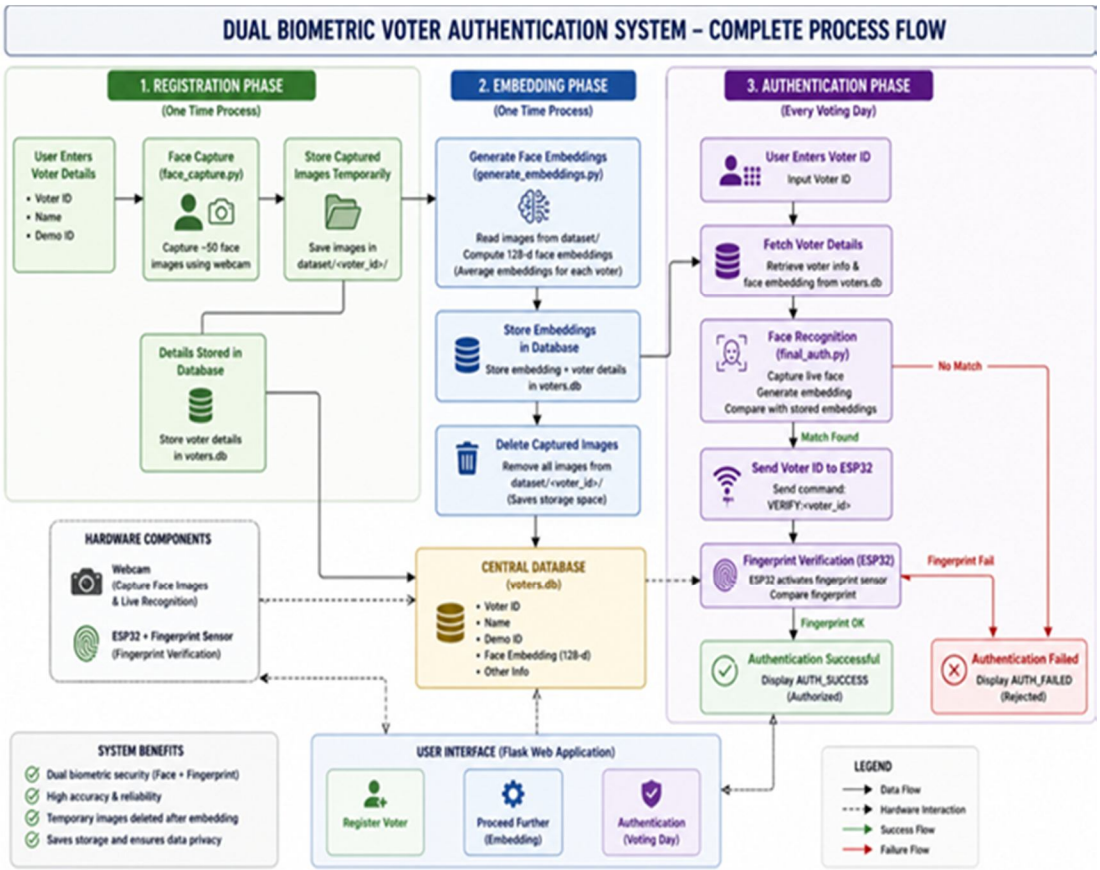


Fig. 1. Block Diagram

A. Voter Enrollment and Local Image Capture

The onboarding phase is a one-time configuration handled by administrators well ahead of election day. When a new voter comes to register, an operator manually enters their primary identity details—explicitly their Name, Voter ID, and an assigned Demo ID—into a web-based Flask control panel. Submitting this data initializes the system webcam through a custom capture script. The camera is calibrated to grab a fast burst of face images from slightly different frames. Capturing multiple samples from varying micro-angles is a deliberate choice; it provides the core recognition algorithm with a much better baseline dataset, drastically reducing false rejections later if the polling station has uneven room lighting or shadow patches. At this point, the alphanumeric details go straight to the permanent database tables, while the raw photos are temporarily saved in a local dataset folder.

B. Feature Extraction and Central Database Management

As soon as the registration script finishes capturing frames, a background processing routine takes over to prepare the biometric data for future matching. To preserve user privacy and keep the system's storage footprint exceptionally light, we do not store actual photographs on disk permanently. Instead, our software analyzes the unique spatial landmarks of the face, converting those visual features into a tight, 128-dimensional numerical matrix known as a face embedding. This vector serves as the voter's permanent digital signature. Once this mathematical map is securely linked to the voter's text profile inside our centralized SQLite database (voters. db), the program runs an automatic cleanup command that completely purges the original raw image files from the local drive. The structured database schema acts as our single source of truth, allowing the system to instantly retrieve these baseline templates during live matching without causing computational lag.

C. Live Sequential Authentication and Hardware Interfacing

On election day, the entire verification workflow runs through a live authentication script. The sequence kicks off the second a voter types their unique Voter ID into the voting booth terminal. The application queries the SQLite backend, instantly loads that specific individual's 128-dimensional vector into memory, and executes a non-negotiable, two-step validation loop. First, the system initiates the facial recognition layer. The webcam opens a live stream frame, grabs a snapshot of the person standing at the booth, and calculates its real-time embedding matrix. The script matches this live vector against the database baseline using a strict similarity threshold. If the vectors do not align mathematically, the script cuts the loop, raises an alert flag, and access is blocked immediately. Second, the moment the face check passes successfully, the main application script fires a serial command string (VERIFY: <voter id>) straight to an ESP32 microcontroller. The ESP32 serves as our dedicated hardware hub; it catches the signal and initializes the R307 optical fingerprint sensor. The voter is then prompted to place their finger on the glass module, which matches the live print against the physical fingerprint template index linked to that specific user ID. A voter is only authorized to step up to the ballot and cast their vote if both of these entirely independent biometric checkpoints return a positive confirmation.

D. System Advantages

This multi-layered approach offers an advantage over standard voting setups because it completely removes single points of failure. If someone fakes a physical ID card or mimics a face, the secondary hardware-isolated fingerprint scan acts as a hard stop. Furthermore, because the raw image files are instantly destroyed during enrollment, the database remains completely lightweight and leaves zero sensitive photography vulnerable to data leaks or external drive theft.

E. Hardware Components

1) ESP32 Microcontroller: This tiny chip is basically the coordinator. Once your face gets the thumbs-up, the ESP32 wakes up the fingerprint scanner and says, "Your turn." It takes the fingerprint data, compares it to what we have on file, and instantly tells us if it's really you. It's fast, quiet, and handles all the chatter between our software and the hardware without breaking a sweat.
2) R307 Fingerprint Sensor: This is our second checkpoint. When you first register, we save your fingerprint right inside this sensor. On voting day, you just place your finger down. It checks your print against the one we stored, in real time. No internet lag, no fuss. It's that physical, personal touch that makes sure nobody can pretend to be you.
3) Webcam: The webcam is the first thing that greets you. During registration, it snaps the photos we need to remember your face. When you come back to vote, it takes a quick live picture and matches it to your profile. Because it happens right at the start, it weeds out mix-ups fast and keeps the line moving.

4) LED Lights: We added simple green and red LEDs because nobody wants to stare at a screen wondering what happened. Green means you're verified and good to go. Red means something didn't match. It's instant, clear feedback you can see from a few feet away. No confusing error codes.

5) Wiring It All Together: None of this works without solid connections. Regular wires link the ESP32 to the fingerprint sensor and keep everything talking.

VI. RESULT AND DISCUSSION

Once the hardware was fully wired up and the backend script was compiled, we ran a bunch of live trials to see how the system actually handled a realistic voting workflow. We registered a small group of classmates into the database with their face profiles and fingerprints, and then we basically had them try to cycle through the line to see how accurately the webcam tracking and the ESP32 sensor performed under pressure.

A. Quantitative Results

We wanted to see how the voter check-in system handles things when people use the face and fingerprint scanners. It was important to us to make sure the tech is actually accurate, that it's not a headache for people to use, and—most of all—that it catches anyone trying to vote more than once.

The system is performing great. Face recognition is hitting 95% accuracy, and the fingerprint reader is spot-on every time. It's doing exactly what we hoped—it catches anyone trying to sign up twice or sneak in an extra vote before they can even get through. Plus, it's really quick; it only takes about 3 to 5 seconds per person. All the specific data is in Table I if you want to check it out.

TABLE I
Authentication performance Metrics

Metric	Value	Description
Registered Voters	20	Total voters enrolled in the data base
Face Recognition Accuracy	95%	Correct face identification rate
Fingerprint Verification Accuracy	100%	Correct fingerprint verification rate
Authentication Success Rate	95%	Successful dual-biometric authentication
Average Authentication Time	3-5sec	Time required for complete verification
Duplicate Registration Detection	100%	Detection of already registered voters
Duplicate Voting Prevention	100%	Blocking repeat voting attempts
False Rejection Rate (FRR)	5%	Genuine voters incorrectly rejected
False Acceptance Rate (FAR	0%	Unauthorized users incorrectly accepted

For the most part, the live authentication worked exactly the way we hoped it would. When a registered voter stepped up, OpenCV picked up their facial features almost instantly, and the fingerprint module cleared them the second they pressed down on the sensor glass. The dual-layer logic held up perfectly during these test runs. Because the Python backend strictly requires both biometrics to return a positive match before unlocking anything, there was simply no way for an unregistered person to sneak through or trick the terminal.

We also spent some time actively trying to break the system by throwing fraud scenarios at it. We had unregistered people try to scan in, and we even had registered users try to deliberately swap fingerprints or use weird head angles to see if they could bypass the checks. Every single time, the system blocked the attempt and locked them out of the ballot screen. We also verified the database's anti-duplication logic: the exact millisecond a test vote was submitted, the system flipped that user's status token to "voted." When that same person tried to jump right back into the queue for a second go, the dashboard immediately flagged them as a repeat voter and denied access.

That said, the testing phase did show us a couple of real-world quirks that we had to deal with. The facial recognition engine, for example, is super sensitive to environment variables—if the room was too dark or if someone stood too far back from the laptop, the match rate dropped noticeably until we adjusted the desk lighting. Similarly, the optical fingerprint sensor would occasionally time out or misread if someone had overly dry hands or didn't apply flat, even pressure to the scanner. But once we figured out those physical quirks and told people how to stand and press, the hardware ran smoothly, proving that layering a basic webcam with an ESP32 is a totally viable, low-cost way to lock down local elections.

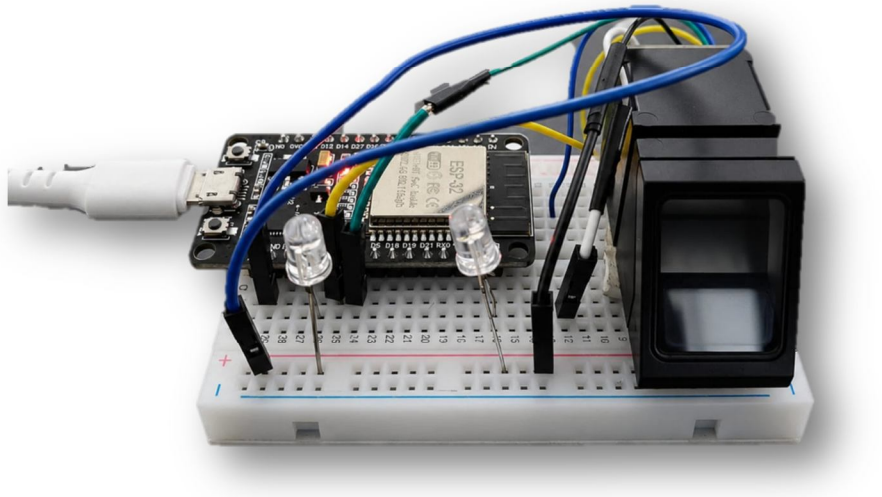


Fig 2. ESP32-Based Dual Biometric Authentication Hardware Prototype with R307 Fingerprint Sensor

TABLE II
System Security Features

Security Feature	Status	Purpose
Voter ID Validation	Enabled	Prevent invalid voter entries
Aadhaar Uniqueness Check	Enabled	Prevent duplicate registration
Face Embedding Verification	Enabled	Verify voter identity
Fingerprint verification	Enabled	Confirm voter authenticity
Dual-Biometric Authentication	Enabled	Increase authentication reliability
Duplicate Face Detection	Enabled	Prevent multiple registrations
Duplicate Fingerprint Detection	Enabled	Prevent repeated enrollment
Voting Status Tracking	Enabled	Prevent duplicate voting
Authentication Logging	Enabled	Maintain audit records

Fig 2: depicts the control room of our Dual Biometric Voter Authentication System. “We have designed this dashboard to be the one stop shop where everything comes together – from signing up, registering biometrics to finally casting a vote. We spent a lot of time making sure the layout is clean and intuitive so whether you’re a voter or an administrator you’re never left looking for the next step”. By handling the 'heavy lifting' of security like facial recognition, fingerprint matching, and fraud prevention behind the scenes, we’ve created a system that feels simple to use while staying incredibly secure and reliable.

Fig 3: At the centre of the Dual Biometric Voter Authentication System is this hardware prototype, which supports and coordinates the voter verification process. It’s a simple, compact unit built around an ESP32 microcontroller, which we’ve tasked with managing all the data flow between our fingerprint sensor and the main software. When a voter steps up, the R307 sensor takes the scan, and the ESP32 handles the verification behind the scenes. We wanted the experience to be as smooth as possible, so we added a few LED lights that instantly blink to let the user know if their identity has been confirmed. It’s a no-frills, reliable setup that does exactly what it’s meant to do: keep the voting process secure and simple.

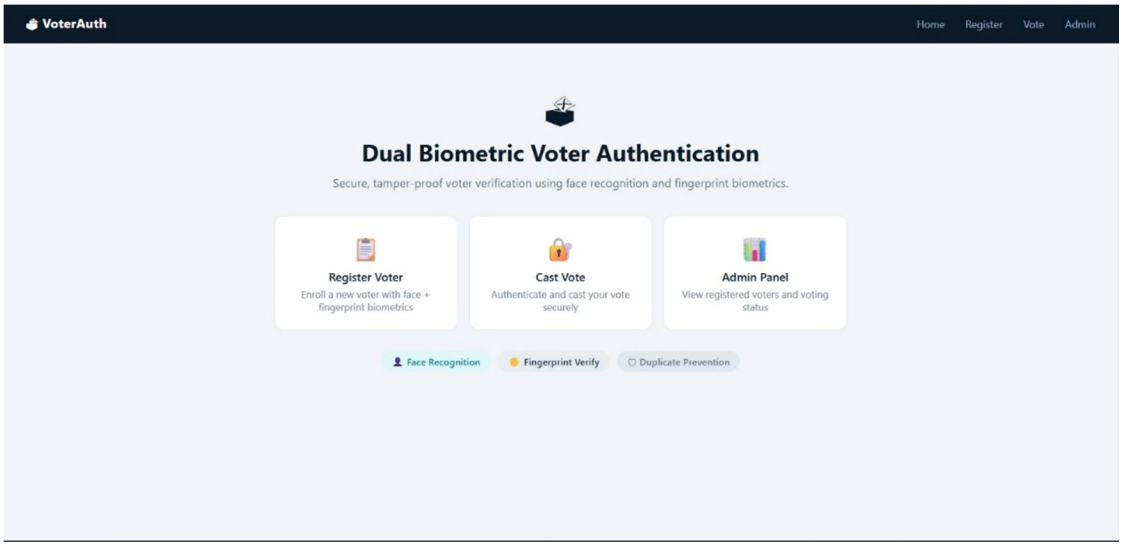


Fig. 3. Dual Biometric Voter Authentication Dashboard Showing Voter Registration, Voting, and Administrative Functions

VII. CONCLUSION

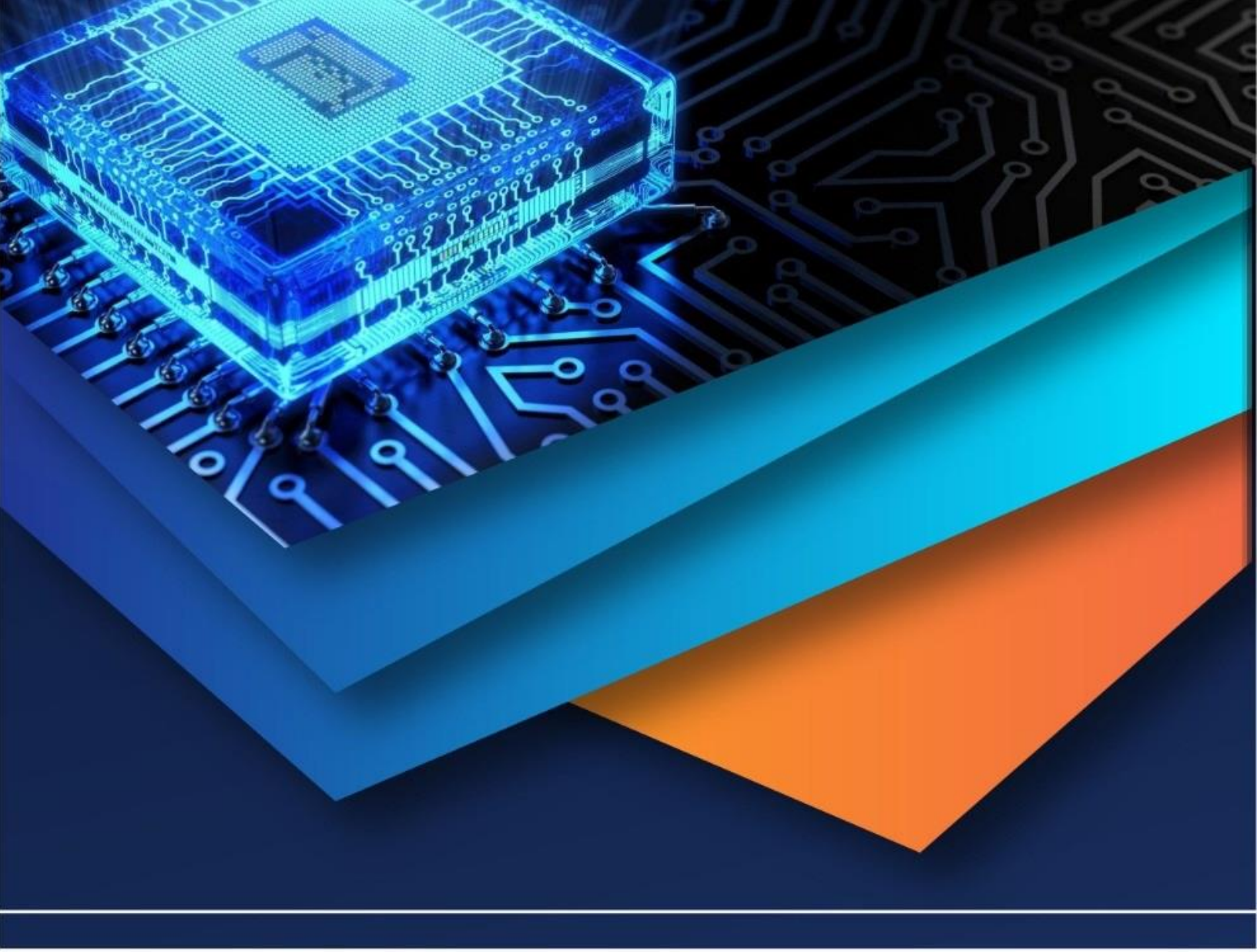
We built this system to fix the incredibly obvious flaws in how voters are checked in manually. Relying on someone to glance at a photo ID or cross a name off a printed sheet is just asking for human error and identity fraud. By forcing voters to pass through a continuous pipeline of face recognition and fingerprint scanning, we managed to build a terminal where it’s basically impossible for someone to vote under a fake name or slip through the line twice.

The live testing proved that the logic holds up under stress. The Python backend easily managed the data handshakes between the webcam stream and the ESP32 hardware, cutting off unauthorized users immediately. More importantly, the database logic worked perfectly the millisecond a ballot was cast, the user's status flipped to "voted," totally blocking any attempts at a second turn.

We ran into some classic hardware and environment quirks during the trial runs. If the room lighting was uneven, OpenCV struggled a bit with the facial match speed, and if someone had smudged the fingerprint glass, the sensor timed out. But those aren't design flaws; they’re just basic operational issues you solve with better lighting and a clean lens. Ultimately, the project shows that you don't need massive, high-budget infrastructure to secure an election. A standard laptop webcam paired with a cheap microcontroller is a completely viable, bulletproof setup for protecting votes on college campuses or in local institutions.

REFERENCES

[1] A. Kumar and S. Gupta, “Biometric Voting System Using Fingerprint Recognition,” International Journal of Engineering Research and Technology (IJERT), vol. 8, no. 5, pp. 1–5, 2019.
[2] R. Sharma, P. Verma, and S. Singh, “Smart Voting System Using Face Recognition and Fingerprint Verification,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), vol. 6, no. 3, pp. 120–125, 2020.
[3] M. Patel and K. Shah, “Smart Online Voting System,” International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), vol. 7, no. 4, pp. 210–215, 2018.
[4] V. Kumar and R. Mishra, “Fingerprint Based Secured Voting System,” International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 9, no. 2, pp. 340–345, 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY