STM JOURNALS

International Journal of
VLSI Circuit Design & Technology

Volume 3, Issue 1, 2025
January–June
DOI (Journal): 10.37591/IJVCDT

https://journals.stmjournals.com/ijvcdt

Research IJVCDT

VHDL Programming for Secure Bootloaders in IoT Security

Kazi Kutubuddin[1,*], Sayyad Liyakat[2]

Abstract

The proliferation of Internet of Things (IoT) devices has brought unparalleled connectivity and convenience. However, this increased connectivity comes with a heightened risk of security vulnerabilities. One of the most critical areas for securing IoT devices is the bootloader, the software responsible for initializing the device and loading the operating system. A compromised bootloader can lead to a complete takeover of the device, making it a prime target for malicious actors. This study explores the role of VHDL (VHSIC Hardware Description Language) programming in creating robust and secure bootloaders for IoT devices, drawing from the key findings and conclusions of research in this domain. Traditional software-based bootloaders, often written in C or assembly language, are susceptible to a range of attacks including buffer overflows, code injection, and reverse engineering. These vulnerabilities can be exploited to install malicious firmware, bypass security mechanisms, and ultimately compromise the entire device. To mitigate these threats, a different strategy utilizes the built-in security benefits of hardware. Implementing secure bootloaders in hardware, using a Hardware Description Language like VHDL, offers a crucial "root of trust". This means that the boot process is anchored in hardware, making it significantly more difficult for attackers to tamper with the initial stages of device startup. This research focuses on leveraging VHDL programming to develop secure bootloaders for IoT devices. Given the vulnerability of software-based bootloaders, this approach aims to establish a hardware root of trust, enhancing security by implementing key functionalities like secure key storage, hardware-enforced authentication, and tamper resistance directly in hardware. The study explores the design principles, challenges, and potential benefits of VHDL-based secure bootloaders, ultimately contributing to a more robust and secure IoT ecosystem.

Keywords: VHDL, IoT, security, bootloaders, malicious

INTRODUCTION

IoT plays a crucial role in our daily lives, spanning from smart homes to industrial automation. However, as IoT devices become more interconnected and complex, new security concerns arise. One major challenge is safeguarding the integrity and authenticity of a device's firmware during the boot process. This is where secure bootloaders come into play, and VHDL programming can help in implementing secure bootloaders in IoT devices [1–3]. A bootloader is a lightweight program that executes upon powering up a device, responsible for loading the primary firmware into memory. A secure bootloader ensures the firmware's authenticity and integrity before initiating the loading process. This ensures that the device only runs legitimate and unaltered firmware, protecting it from malware and unauthorized modifications [4–8]. VHDL (Very High-Speed Integrated Circuit Hardware Description Language) is a programming

***Author for Correspondence**
Kazi Kutubuddin
E-mail: drkkazi@gmail.com

[1]Professor and Head, Department of Electronics and Telecommunication Engineering, Brahmdevdada Mane Institute of Technology, Solapur, Maharashtra, India
[2]Student, Electronics and Telecommunication Engineering, Brahmdevdada Mane Institute of Technology, Solapur, Maharashtra, India

Received Date: March 28, 2025
Accepted Date: March 29, 2025
Published Date: April 09, 2025

Citation: Kazi Kutubuddin, Sayyad Liyakat. VHDL Programming for Secure Bootloaders in IoT Security. International Journal of VLSI Circuit Design & Technology. 2025; 3(1): 19–28p.

language used to describe and design electronic circuits, including Field-Programmable Gate Arrays (FPGAs). VHDL programming can be used to implement secure bootloaders in IoT devices by designing custom hardware circuits that perform secure boot operations [9, 10].

Here are some ways VHDL programming can be used for secure bootloaders:

1. *Hardware Root of Trust:* VHDL (Very High-Speed Integrated Circuit Hardware Description Language) is a programming language used to describe and design electronic circuits, including Field-Programmable Gate Arrays (FPGAs). VHDL programming can be used to design a hardware root of trust that stores cryptographic keys and performs secure boot operations.
2. *Secure Hash Algorithms:* VHDL programming can be used to implement secure hash algorithms, such as SHA-256, that are used to verify the integrity of the main firmware. The secure hash of the main firmware can be compared with a stored hash value during the boot process.
3. *Cryptographic Algorithms:* VHDL programming can also be used to implement cryptographic algorithms, such as AES, that are used to encrypt and decrypt cryptographic keys and messages. This can be used to securely transfer cryptographic keys between the secure bootloader and the main firmware.
4. *Secure Communication:* VHDL programming can be used to implement secure communication protocols, such as SSL/TLS, that are used to secure communication between the secure bootloader and the main firmware. This can be used to prevent attackers from intercepting and tampering with cryptographic keys and messages.
5. *Secure Storage:* VHDL programming can be used to design secure storage solutions, such as encrypted flash memory, that are used to store cryptographic keys and sensitive data. This can be used to protect cryptographic keys and sensitive data from physical attacks and unauthorized access.

VHDL programming enables the development of secure bootloaders for IoT devices, adding an extra layer of protection against malware and unauthorized alterations. By designing custom hardware circuits that perform secure boot operations, VHDL programming can help ensure the integrity and authenticity of the device's firmware, protecting IoT devices from security threats.

When designing secure bootloaders with VHDL programming, it is essential to follow best practices for secure coding and hardware design. This includes ensuring the confidentiality, integrity, and availability of the secure bootloader, as well as implementing appropriate security measures to prevent physical and remote attacks [11–15].

Overall, VHDL programming for secure bootloaders is a powerful tool for IoT security, providing a foundation for secure firmware updates and protecting IoT devices from security threats.

HOW VHDL PROGRAMMING FORTIFIES SECURE BOOTLOADERS IN IOT SECURITY

The rapid growth of IoT in recent years has linked billions of devices, from smart home gadgets to industrial sensors. However, this widespread connectivity also introduces major security risks, as each device can serve as a potential access point. Safeguarding these systems is crucial to maintaining their integrity. One crucial defense mechanism lies within the secure bootloader, and its robust implementation often hinges on the power and precision of VHDL (VHSIC Hardware Description Language) programming.

The bootloader is the initial code that runs when an IoT device is powered on, acting as the system's gatekeeper. Its main function is to set up the hardware and load the operating system (OS) or application software. A secure bootloader goes a step further by verifying the authenticity and integrity of the software image before allowing the device to boot. This prevents malicious actors from installing compromised or malicious software by exploiting vulnerabilities in the boot process [16–19].

Why Secure Bootloaders are Essential for IoT Security

- *Protection Against Malware:* Prevents the execution of unauthorized code, effectively blocking malware from infiltrating the system.
- *Prevention of Firmware Tampering:* Ensures that only genuine, manufacturer-approved firmware can be loaded, preventing unauthorized modifications.
- *Protection of Sensitive Data:* By verifying the integrity of the software, it safeguards sensitive data stored within the device from being compromised by malicious code.
- *Compliance with Regulatory Standards:* Secure bootloaders are frequently mandated to meet security regulations and compliance standards.

While software-based secure bootloaders exist, hardware-based solutions, often implemented using FPGAs or ASICs, offer a significantly higher level of security. This is where VHDL shines. VHDL allows developers to precisely define the hardware logic and behavior of the secure bootloader, ensuring it operates as intended and is resistant to tampering [20, 21].

Here is how VHDL contributes to the security of bootloaders

- *Hardware-Level Control:* VHDL enables designers to create custom hardware logic that is inherently more difficult to compromise compared to software-based solutions. This granular control allows for the implementation of secure cryptographic algorithms and secure key storage directly in hardware.
- *Deterministic Behavior:* VHDL allows for the creation of bootloaders with highly predictable and deterministic behavior. This removes uncertainty and minimizes the risk of weaknesses that attackers could take advantage of.
- *Customizable Security Features:* VHDL allows developers to tailor the secure bootloader to specific hardware and security requirements. This degree of personalization is essential for meeting the varied requirements of different IoT devices.
- *Hardware Security Modules (HSMs) Implementation:* VHDL enables the creation of specialized hardware security modules within the bootloader, ensuring secure storage and management of cryptographic keys. This separation of essential security tasks strengthens the system's overall protection.
- *Physical Attack Resistance:* Hardware-based secure bootloaders implemented in VHDL can be designed with countermeasures against physical attacks, such as side-channel analysis and fault injections, making them more resilient to sophisticated threats.

Key VHDL Implementation Considerations for Secure Bootloaders

- *Cryptographic Algorithm Implementation:* VHDL is used to implement cryptographic algorithms like AES, SHA, and RSA in hardware, providing secure authentication and integrity checks of the firmware image.
- *Secure Key Management:* Ensuring secure storage and management of keys within hardware is essential. VHDL allows developers to create secure memory regions within the FPGA or ASIC to protect cryptographic keys from unauthorized access.
- *Hardware Root of Trust (HRoT):* A VHDL-based Hardware Root of Trust (HRoT) establishes a reliable and secure base for the entire boot sequence. It acts as the ultimate authority, verifying the integrity of the bootloader itself before allowing it to execute.
- *Tamper Detection and Response:* VHDL allows developers to implement hardware-based tamper detection mechanisms that can detect physical tampering attempts and trigger appropriate responses, such as erasing sensitive data or disabling the device.

While VHDL offers significant advantages in building secure bootloaders, challenges remain. Complexity, development time, and verification are crucial considerations. Furthermore, the evolving landscape of IoT threats demands continuous innovation.

Future directions in VHDL programming for secure bootloaders include

- *Formal Verification Techniques:* Using formal verification tools to rigorously validate the accuracy and security of VHDL code through mathematical proof.
- *Standardized Security IPs:* Developing reusable, pre-verified VHDL-based security IPs to accelerate secure bootloader development.
- *Integration with Security Frameworks:* Integrating VHDL-based secure bootloaders with established security frameworks and standards.
- *Lightweight Cryptography:* Exploring and implementing lightweight cryptographic algorithms in VHDL that are optimized for resource-constrained IoT devices.

The secure bootloader plays a vital role in safeguarding the security framework of IoT devices. VHDL programming plays a vital role in building robust, hardware-based secure bootloaders that are resistant to a wide range of threats. As the IoT continues to grow and evolve, so too will the importance of VHDL in ensuring the security and integrity of these interconnected devices. By leveraging the power and flexibility of VHDL, developers can create the silent guardians that protect the IoT ecosystem from malicious attacks [22–26].

DESIGNING SECURE BOOTLOADERS IN IOT WITH VHDL

IoT promises a connected world, but this connectivity comes with inherent security risks. A vulnerable IoT device can serve as an entry point for cybercriminals, allowing them to steal sensitive information, disrupt essential systems, and execute widespread attacks. One of the most vulnerable points in an IoT device is the bootloader, the critical piece of software responsible for initializing the hardware and loading the operating system. If compromised, it can allow an attacker to install malicious code that is executed before any security mechanisms are even active. This study explores the design of secure bootloaders using VHDL (Very High-Speed Integrated Circuit Hardware Description Language) as a means to build a robust and trustworthy foundation for IoT security.

The bootloader serves as the device's initial layer of protection. It controls the initial configuration of the hardware, checks the integrity of the flash memory, and loads the main application code. A secure bootloader guarantees that only verified and authorized firmware is installed on the device, safeguarding it from malicious code infiltration. Its primary functions include:

- *Authentication:* Ensures the firmware image is genuine and has not been altered before execution.
- *Integrity Verification:* Identifies any unauthorized modifications or corruption in the firmware.
- *Secure Key Storage:* Safeguards cryptographic keys essential for authentication and encryption.
- *Rollback Protection:* Blocks the installation of outdated firmware versions that may have security vulnerabilities.
- *Secure Update Mechanism:* Facilitates safe and dependable firmware updates while in operation.

While software-based bootloaders exist, a hardware-implemented bootloader offers several advantages in terms of security and performance. VHDL, a hardware description language, allows for the precise and deterministic implementation of these critical security functionalities. Here is why VHDL is a suitable choice:

- *Hardware-Level Control:* VHDL allows designers to manipulate hardware resources directly, enabling fine-grained control over security-critical operations like memory access and cryptographic calculations.
- *Tamper Resistance:* Hardware implementations are inherently more tamper-resistant than software, making it more difficult for attackers to modify or bypass security mechanisms.
- *Performance:* Hardware acceleration enhances the efficiency of cryptographic processes, leading to faster boot times and improved system responsiveness.
- *Formal Verification:* VHDL designs can be formally verified, ensuring that the implemented security mechanisms meet rigorous specifications and are free from vulnerabilities.

- *Customization:* VHDL allows for highly customized bootloader designs tailored to the specific hardware and security requirements of the IoT device.

The design shown in Figure 1, of a secure bootloader in VHDL involves several key components and considerations:

1. *Hardware Platform Abstraction:* Develop VHDL modules to abstract the underlying hardware platform, including memory controllers, peripherals, and cryptographic accelerators. This allows for portability and easier adaptation to different hardware architectures.
2. *Cryptographic Implementation:* Implement cryptographic algorithms for authentication and integrity verification in VHDL. This can include:
   - *Hashing Algorithms:* SHA-256 or SHA-3 for generating cryptographic hashes of the firmware image.
   - *Digital Signatures:* RSA or Elliptic Curve Cryptography (ECC) for authenticating the firmware image using a digital signature.
   - *Symmetric Encryption:* AES is used to encrypt the firmware image, ensuring protection against unauthorized access.
3. *Secure Key Management:* Implement a secure key storage mechanism using dedicated hardware registers or on-chip flash memory. Protect these keys from unauthorized access using access control mechanisms.
4. *Authentication and Integrity Verification Module:* This component ensures the firmware image's authenticity and integrity during the boot process. It calculates the hash of the firmware image and compares it against a trusted hash stored in secure memory. It authenticates the firmware image by validating its digital signature with the stored public key.
5. *Secure Update Mechanism:* Implement a secure update mechanism that allows for updating the firmware in the field without compromising security. This typically involves encrypting the firmware update package and securely transmitting it to the device.
6. *Rollback Protection:* Implement a rollback protection mechanism to prevent the installation of older, potentially vulnerable firmware versions. This can be achieved by storing a version number in secure memory and validating that the new firmware version is greater than the current version.

Example: A Simplified VHDL Module for SHA-256 Hash Calculation

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity sha256_core is
   port (
      clk       : in  std_logic;
      rst       : in  std_logic;
      data_in   : in  std_logic_vector(511 downto 0);  -- 512-bit input block
      start     : in  std_logic;
      ready_out : out std_logic;
      hash_out  : out std_logic_vector(255 downto 0)   -- 256-bit hash output
   );
end entity;

architecture rtl of sha256_core is
   -- Internal signals and logic for SHA-256 algorithm
   -- (Simplified representation - actual SHA-256 requires a complex implementation)
   signal internal_hash : std_logic_vector(255 downto 0);
begin
   process (clk, rst)
```



```
    begin
        if rst = '1' then
            internal_hash <= (others => '0');  -- Initialize hash
            ready_out <= '0';
        elsif rising_edge(clk) then
                if start = '1' then
                        -- placeholder -  perform SHA-256 steps using data_in to update internal_hash
                        internal_hash <= (others => '1');  -- replace with actual SHA-256 logic
                        ready_out <= '1';
                    else
                        ready_out <= '0';
            end if;
        end if;
    end process;

    hash_out <= internal_hash;
end architecture;
```

Note: This is a highly simplified example and does not represent a fully functional SHA-256 implementation. A real implementation would require significantly more complex logic and would need to adhere to the SHA-256 standard.

Designing a secure bootloader with VHDL is not without its challenges:

- *Complexity:* Implementing cryptographic algorithms and security protocols in hardware can be complex and requires a deep understanding of hardware design.
- *Verification:* Thoroughly verifying the security of the bootloader design is essential, but it can be time-consuming and resource-intensive.
- *Cost:* Hardware implementations can be more expensive than software implementations due to the cost of development tools, hardware components, and testing.
- *Power Consumption:* Designing an energy-efficient bootloader is essential for extending battery life in IoT devices.
- *Evolving Threats:* Security threats are always changing, making it essential to continuously monitor and update the bootloader design.

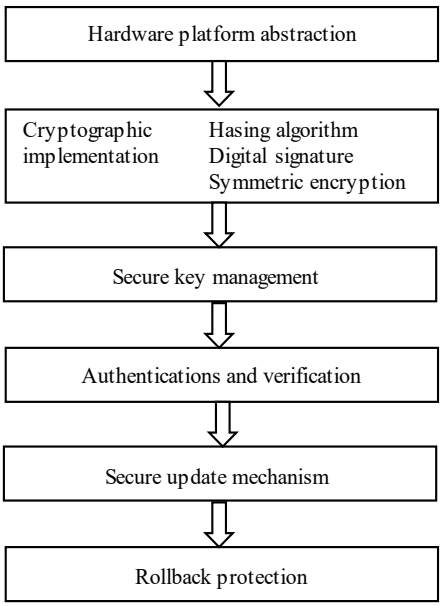

Figure 1. Key components of bootloader security.

Designing secure bootloaders using VHDL provides a robust and reliable foundation for IoT security. By leveraging the advantages of hardware-based security, developers can build trust into the very core of their devices. While challenges exist, the benefits of using VHDL for secure bootloader design, including tamper resistance, performance, and hardware-level control, outweigh the complexities. As the IoT continues to grow, securing the bootloader with a carefully designed and implemented VHDL solution is essential for protecting connected devices from malicious attacks and ensuring a safer and more secure future.

DISCUSSION

The rapid growth of IoT has connected billions of devices to the internet, but this expansion also brings major security risks. A key vulnerability lies in the bootloader, the first software that runs when a device starts up. A compromised bootloader can give attackers complete control over the device, jeopardizing the entire system. Consequently, security-conscious developers are increasingly turning to hardware-based security solutions written in languages like VHDL (VHSIC Hardware Description Language) to fortify their IoT bootloaders.

VHDL is a hardware description language used to design and simulate digital circuits. Unlike software languages like C or Python, VHDL allows developers to specify the precise behavior of hardware components. This granular control offers several advantages when building secure bootloaders for IoT devices:

- *Hardware-Level Security:* VHDL code is ultimately implemented in hardware, making it resistant to many software-based attacks. Features like secure key storage, cryptographic acceleration, and immutable boot code can be embedded directly into the silicon, making them far more difficult to tamper with.
- *Enhanced Trust:* By designing security-critical components in VHDL, developers gain a deeper understanding of the underlying hardware behavior. This transparency increases trust in the integrity of the boot process.
- *Tamper Resistance:* Implementing security features in hardware makes it significantly harder for attackers to bypass them. Techniques like Physically Unclonable Functions (PUFs) and secure memory regions can be integrated directly into the FPGA or ASIC, providing strong tamper resistance.
- *Performance and Efficiency:* VHDL allows for highly optimized hardware designs. Specialized hardware accelerators for cryptographic tasks can greatly enhance boot speed and minimize power usage, which are essential factors in IoT environments with limited resources.
- *Predictability and Verification:* VHDL enables thorough simulation and verification of hardware designs before implementation. This enables developers to detect and address security flaws early in the development process, minimizing the risk of exploitation.

While VHDL offers significant advantages, successfully implementing a secure bootloader requires careful planning and execution. Here are some key considerations:

- *Threat Modeling:* Identifying potential attack vectors is crucial. Consider threats like bootloader tampering, firmware downgrading, and side-channel attacks.
- *Secure Key Management:* Safeguarding and efficiently handling cryptographic keys is of utmost importance. Hardware-based solutions like secure elements and Trusted Platform Modules (TPMs) can be integrated into the VHDL design.
- *Cryptographic Algorithms:* Choosing robust and well-vetted cryptographic algorithms is essential. Standards such as AES for encryption, SHA-256 for hashing, and ECC for digital signatures ensure robust security in modern cryptographic systems.
- *Secure Boot Process:* Ensure a secure boot process by validating the integrity of the bootloader and firmware images before they are executed. This typically involves using digital signatures and hash algorithms.

- *Rollback Protection:* Implement mechanisms to prevent attackers from downgrading the firmware to vulnerable versions.
- *Side-Channel Attack Mitigation:* Design hardware components to mitigate side-channel attacks, such as power analysis and timing attacks. This may include methods such as concealing and obscuring.
- *Code Review and Auditing:* Conduct a comprehensive review and assessment of the VHDL code to detect potential weaknesses and verify compliance with security best practices.
- *Hardware Verification:* Rigorously test the hardware implementation to verify its security and functionality.

Here are some examples of how VHDL can be used to implement security features in IoT bootloaders:

- *Hardware Root of Trust:* Implementing a hardware root of trust using VHDL allows for secure storage of keys and cryptographic parameters.
- *Cryptographic Accelerators:* Implementing dedicated hardware accelerators for cryptographic operations, such as AES or SHA-256, in VHDL can significantly improve boot times and reduce power consumption.
- *Secure Boot Verification:* Implementing the secure boot process in VHDL ensures that only authenticated and authorized firmware is loaded.
- *PUF Implementation:* Implementing a Physically Unclonable Function (PUF) in VHDL can provide a unique hardware fingerprint for device identification and authentication.

As IoT devices become more advanced and interconnected, the need for strong security measures will keep increasing. VHDL, with its ability to create hardened, customized hardware security features, is poised to play a vital role in securing IoT bootloaders and protecting devices from malicious attacks. By understanding the advantages and challenges of VHDL-based security, developers can build more secure and resilient IoT systems. The shift towards hardware-based security, facilitated by languages like VHDL, is a critical step in building a more trusted and secure IoT ecosystem.

In conclusion, VHDL offers a powerful and flexible platform for building secure bootloaders in IoT devices. By leveraging its hardware-level control and security features, developers can create robust defenses against a wide range of attacks, ultimately strengthening the overall security of the IoT ecosystem. As the threat landscape evolves, VHDL will continue to be a crucial tool in the fight to protect connected devices.

CONCLUSION

Implementing secure bootloaders in VHDL offers a significant advantage in mitigating the risks of compromised IoT devices. By establishing a hardware root of trust, VHDL enables developers to create bootloaders that are more resistant to tampering, code injection, and other common attacks. While challenges such as increased complexity and development time exist, the security benefits of hardware-based solutions often outweigh the drawbacks, particularly for critical IoT applications where security is paramount. Future research should focus on developing standardized VHDL modules for secure bootloaders and exploring hybrid approaches that combine the advantages of both hardware and software security mechanisms, further solidifying the foundation for a secure and trustworthy IoT landscape. Ultimately, the adoption of VHDL-based secure bootloaders represents a crucial step towards building a more secure and resilient IoT ecosystem.

REFERENCES

1. Jadhav M, Nerkar PM. FPGA-Based finger vein recognition system for personal verification. Int J Eng Res Gen Sci. 2015; 3(4): 382–8.
2. Kulkarni S, Nerkar PM. Retina Image Decomposition Using Variational Mode Decomposition. Int Res J Eng Technol. 2018 Jun; 5(06): 2510–2512.

3. Khadake S, Kawade S, Moholkar S, Pawar M. A review of 6G technologies and its advantages over 5G technology. In: Techno-Societal 2016, International Conference on Advanced Technologies for Societal Applications. Cham: Springer International Publishing; 2022 Dec 9; 1043–1051.
4. Patil VJ, Khadake SB, Tamboli DA, Mallad HM, Takpere SM, Sawant VA. Review of AI in power electronics and drive systems. In 2024 IEEE 3rd International conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC). 2024 Feb 23; 94–99.
5. Dudgikar AB, Ingalgi AA, Jamadar AG, Swami OR, Khadake SB, Moholkar SV. Intelligent battery swapping system for electric vehicles with charging stations locator on IoT and cloud platform. Int J Adv Res Sci Commun Technol. 2023 Jan; 3(1): 204–8.
6. Wamborikar YS, Sinha A. Solar powered vehicle. In proceedings of the World Congress on Engineering and Computer Science. 2010 Oct 20; 2: 20–22.
7. Zohuri B, Mossavar-Rahmani F. The symbiotic relationship unraveling the interplay between technology and artificial intelligence (an intelligent dynamic relationship). Journal of Energy and Power Engineering. 2023 Apr; 17(2): 63–68.
8. Khattak MA. PLC based intelligent traffic control system. Int J Electr Comput Sci. 2011 Dec; 11(06): 69–73.
9. Samy CK, Ahmadi HB, Atfah YA, Dol SS, Alavi M. Design of portable vortex bladeless wind turbine: The preliminary study. J Adv Res Appl Mech. 2023 Feb; 102(1): 32–43.
10. Abass N, Vaidya R, Satav R, Jeyavel J. Automatic sanitizer dispenser with temperature screening. International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT). 2021; 7(3): 283–5.
11. Dhawale A, Kamble A, Ghule S, Gawande M. Solar Powered Indoor Air Purifier With Air Quality Monitor. In 2023 IEEE 9th International Conference on Electrical Energy Systems (ICEES). 2023 Mar 23; 206–209.
12. Shabnam S, Latha HN. Design and implementation of saliency detection model in h. 264 standard. Int J Sci Res. 2014; 3(6): 2014–20.
13. Mane A, Kharade M, Sonkambale P, Tapase S, Kudte SS. Design & analysis of vortex bladeless turbine with gyro e-generator. Int J Innov Res Sci Eng. 2017 Apr; 3(4): 445–52.
14. Muniappan A, Thiagarajan C, Kumar GA, Joseph Raj X, Irene J, Niranjan N. Conversion of Conventional Vehicle Into Solar Powered Electric Vehicle—A Realistic Approach. Int J Innov Res Sci Eng Technol. 2014; 3(9): 16232–7.
15. Pawar P, Gite N, Sonawane M. Automatic Load Sharing of Transformer By using PLC. Int Res J Eng Technol. 2020; 7(7): 339–42.
16. Nikhil KN. A review on battery management system for electric vehicles. Int J Res Appl Sci Eng Technol. 2022 Jul; 10(7): 3699–3710.
17. Chounde A, Gopnarayan BB, Patil KB, Kamble SS. Human Health Care System: A New Approach towards Life. Grenze Int J Eng Technol. 2024 Jun 15; 10(2): 5487–5494.
18. Patil VJ, Mallad HM, Gopnarayan BB, Pati KB. Maximize Farming Productivity through Agriculture 4.0 based Intelligence, with use of Agri Tech Sense Advanced Crop Monitoring System. Grenze Int J Eng Technol. 2024 Jun 15; 10(2): 5127–5134.
19. Shrivastava K, Bansal R, Jain H, Doshi N, Soni N, Soni N. Conversion of conventional vehicle into an electric vehicle. Advances and Applications in Mathematical Sciences. 2020;20(1):15–24.
20. Atabani AE. Biodiesel: a promising alternative energy resource. Alternative Fuels Research Progress. International Energy and Environmental Foundation Publishers; India. 2013; 93.
21. Kazi KS. AI-Driven-IoT (AIIoT)-Based Decision Making in Kidney Diseases Patient Healthcare Monitoring: KSK Approach for Kidney Monitoring. In: AI-Driven Innovation in Healthcare Data Analytics. IGI Global Scientific Publishing; Pennsylvania, United States. 2025; 277–306.
22. Liyakat KK. Smart agriculture based on AI-driven-IoT (AIIoT): A KSK approach. Advance Research in Communication Engineering and its Innovations. 2024; 1(2): 23–32.
23. Kazi KS. KK Approach to Increase Resilience in Internet of Things: A T-Cell Security Concept. In: Analyzing Privacy and Security Difficulties in Social Media: New Challenges and Solutions. IGI Global Scientific Publishing; 2025; Pennsylvania, United States. 87–120.

24. Alaparthy V. A Study on the Adaptability of Immune System Principles to Wireless Sensor Network and IoT Security. Dissertation. Florida, United States: University of South Florida; 2018.
25. Thakur A, Kumar A. Diagnosing of Disease Using Machine Learning in Internet of Healthcare Things. In: Practical Artificial Intelligence for Internet of Medical Things. CRC Press; Florida, United States. 2023 Feb 28; 241–262.
26. Vo QM, Cao NT, Ton-That AH. Unsafe image classification using convolutional neural network for brand safety. In 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE). 2020 Dec 16; 1–4.

