International Journal of Engineering and Techniques - Volume 12 Issue 3, May-June - 2026

Comparative Analysis of Arduino Microcontroller and Digital Logic Gates for Small Embedded System Applications

Punith Kumar D[1]
1(Department of Electrical and Communication Engineering, Indian Institute of Information Technology, Sricity
Email: punithpuni4137@gmail.com)

Abstract:

This paper explores the critical design choices in embedded systems by comparing microcontroller-based architectures, like Arduino, against traditional digital logic gate designs. While microcontrollers offer incredible software flexibility and streamlined hardware for handling complex tasks, logic gate systems provide unmatched speed and minimal power consumption by processing functions directly through physical circuitry. To practically evaluate these tradeoffs, both approaches were implemented and tested across three distinct applications: a digital dice, a water level indicator, and a password verification system. By analyzing key performance metrics—specifically power consumption, operational speed, cost, hardware complexity, and overall flexibility—this study delivers a clear, practical guide to help engineers select the optimal architecture for their specific design goals.

Keywords — **Arduino, Digital logic design, Embedded systems, Water level Indicator, Digital Dice, Password checker.**

INTRODUCTION

Embedded systems play an important role in modern electronics, enabling automation and control in various small scale applications. Such systems can be developed using different approaches, among which microcontroller based systems and digital logic gate circuits are widely used. Digital logic gates provide hardware-based implementation with dedicated functionality, while microcontrollers such as Arduino offer programmable and flexible solutions for embedded applications.

Choosing the appropriate approach for small embedded systems, depends on multiple factors, including hardware requirements, complexity, performance, and cost. While digital logic gate circuits are often suitable for fixed operations, Arduino-based systems provide easier modification and expansion through software programming.

This paper presents a comparative analysis of Arduino microcontroller-based circuits and digital logic gate based systems using three embedded applications: Water Level Indicator, Password Checker, and Digital Dice**.** The comparison is carried out based on key parameters such as components used, scalability, flexibility, circuit complexity, output delay, and circuit cost**.** Through this study, the research aims to identify the advantages and limitations of both approaches and determine their suitability for small embedded system applications.

LITERATURE REVIEW

Recent advancements in embedded systems have increased the use of both microcontroller based and digital logic gate based implementations in electronic applications. Arduino microcontrollers have gained significant popularity due to their low cost, programmability, and ease of integration into small embedded systems. Studies have highlighted Arduino as a flexible platform for rapid prototyping and automation applications, making it widely adopted in education, industrial systems, and IoT-based projects. Research has shown that Arduino-based systems simplify hardware implementation while allowing easy modification through software programming.

Traditional digital logic gate systems, on the other hand, have been widely used in electronics for implementing fixed and dedicated operations. Logic gates such as AND, OR, NAND, NOR, and XOR form the foundation of digital electronic systems and are known for their fast response and reliable hardware-level execution. Several studies emphasize the importance of logic gate-based circuits in embedded electronics due to their deterministic operation and minimal software dependency.

Previous research has also explored the relationship between Arduino and digital logic systems. Some studies demonstrated the use of Arduino for simulating or testing digital logic gates and integrated circuits, showing that microcontrollers can reproduce logic behaviour while reducing hardware complexity. These works suggest that Arduino offers higher flexibility and easier implementation compared to traditional gate-level circuits in certain applications.

However, most existing studies focus on either Arduino-based implementations or digital logic gate systems individually, with limited practical comparative analysis between the two approaches using real embedded applications. In particular, there is a lack of comparative evaluation based on practical parameters such as components used, scalability, flexibility, circuit complexity, output delay, and circuit cost**.** Therefore, this research aims to bridge this gap through a comparative study of Water Level Indicator, Password Checker, and Digital Dice circuits implemented using both Arduino microcontrollers and digital logic gates.

METHODOLOGY

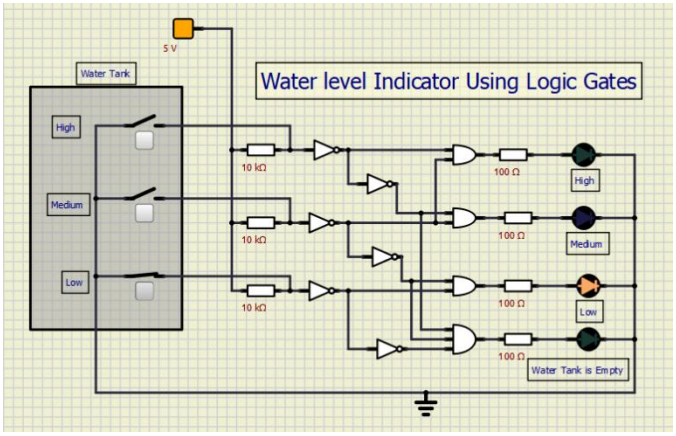
This study follows a comparative experimental methodology to evaluate the performance and implementing differences between Arduino-based systems and traditional logic gate systems. The objective of this research is to analyze how both approaches differ in terms of scalability, flexibility, circuit complexity, output delay, and implementation cost.

To perform the comparison, identical electronic applications were developed using both Arduino and logic gate approaches. The performance of each implementation was then analyzed using predefined evaluation parameters.

1. Water Level Indicator

The objective of the Water Level Indicator system is to monitor and indicate the level of water present in a tank using both digital logic gates and an Arduino microcontroller approach. The circuit is designed to detect different water levels Low, Medium, High, and Empty and provide corresponding visual indications through LEDs.

Working of Water Level Indicator Using Logic Gates



The logic gate-based water level indicator operates using a combination of NOT gates and AND gates to detect the presence or absence of water at different levels inside the tank. Water probes are placed at Low, Medium, and High positions within the tank. These probes act as input signals to the logic circuit.

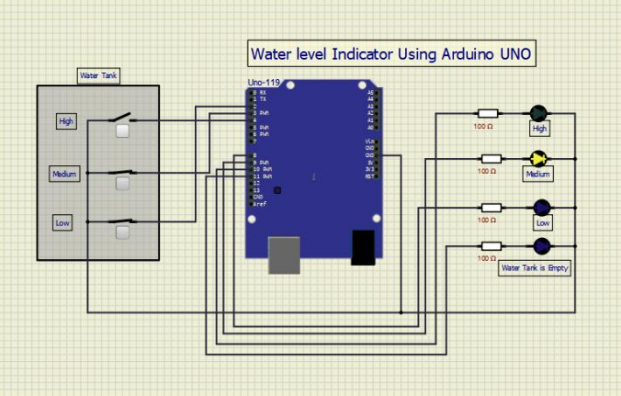
When water reaches a particular probe, the electrical conductivity of water creates a connection, generating an input signal. Based on the combination of active probes, the logic gate network processes the signals and activates the corresponding LED indicator.

For example, when the water level reaches only the lower probe, the Low LED turns ON. As the water rises to the medium probe, the logic conditions

activate the Medium LED, and when the tank reaches the highest level, the High LED is illuminated. If none of the probes detect water, the system indicates that the water tank is empty.

The circuit relies entirely on hardware-level signal processing, where logic gates directly determine the output state without requiring software programming. This results in very fast output response due to nanosecond-level gate switching.

Working of Water Level Indicator Using Arduino UNO



The Arduino-based water level indicator uses an Arduino UNO microcontroller to monitor water levels through sensor probes connected to input pins. Similar to the logic gate system, probes are placed at Low, Medium, and High levels inside the tank.

When water touches a probe, an electrical signal is sent to the Arduino input pin. The Arduino continuously reads these input signals and executes programmed conditions stored in its memory. Based on the detected water level, the microcontroller turns ON the corresponding LED indicators.

For instance, if the lower probe detects water, the Arduino activates the Low LED. As water reaches higher levels, the program updates the output and activates the Medium or High LED accordingly. If no water is detected at any probe, the system activates the Water Tank is Empty indication.

Unlike the logic gate system, the Arduino implementation depends on software programming, making it easier to modify, update, or expand without redesigning the hardware circuit.

Difference Between Logic Gate-Based and Arduino-Based Water Level Indicator

Although both systems perform the same function of detecting and displaying water levels, their implementation methods are significantly different.

The logic gate-based system is purely hardware-oriented and operates using predefined gate combinations. It offers faster response time because signal processing occurs directly through electronic gates without software execution. However, modifying the system or adding additional functionalities requires redesigning the hardware connections, making it less flexible and scalable.

In contrast, the Arduino-based system uses software programming to process water level signals. This approach provides greater flexibility, easier modification, and higher scalability since new features can be added through programming changes rather than hardware redesign. However, the use of a microcontroller introduces a slight increase in output delay due to program execution time and also increases implementation cost.

Thus, the choice between the two systems depends on application requirements. Logic gates are more suitable for simple, fixed-function systems requiring faster response, while Arduino is better suited for applications needing flexibility, scalability, and easier upgrades.

Comparative Analysis of Water Level Indicator		
Parameters	**Logic Gates**	**Arduino UNO**
Components Used	6 NOT gates, 4 AND gates, resistors, LEDs, water probes	Arduino UNO, resistors, LEDs, water probes
Scalability	Low (~30%)	High (~90%)
Flexibility	Low (~20%)	High (~90%)
Circuit Complexity	Moderate to High (~60%)	Moderate (~60%)
Output Delay	~20 ns	~400 ns
Cost Required	₹200–₹300	₹1000–₹1200

Result and Discussion

The comparative analysis of the Water Level Indicator revealed noticeable differences between the two implementation methods. The logic gate-based system demonstrated a significantly faster response time due to direct hardware-level signal processing, achieving output delays in the nanosecond range. Additionally, the implementation cost was comparatively lower because only basic logic gates and electronic components were required.

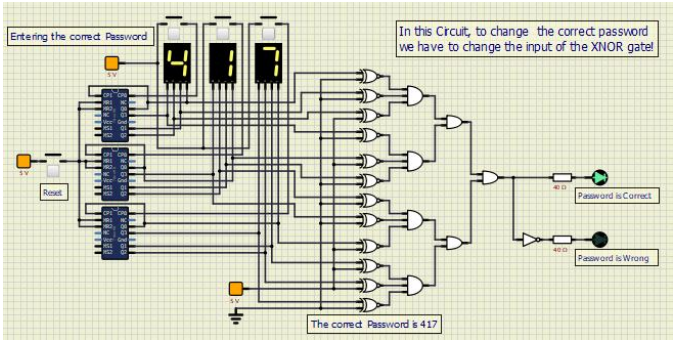
However, the Arduino-based system showed superior performance in terms of scalability and flexibility. Modifications to the system could be easily achieved through software programming without changing the hardware structure. This makes Arduino more practical for applications where future expansion or additional functionalities are required.

Although the Arduino implementation introduced a slight increase in output delay and higher cost, its simplicity in design and ease of implementation made it highly suitable for modern embedded system applications. On the other hand, logic gates proved to be effective for dedicated systems where speed and low cost are the primary requirements.

2. Password checker

The objective of the Password Checker system is to verify whether the entered password matches a predefined password using both digital logic gate-based implementation and an Arduino UNO microcontroller-based implementation. The system is designed to provide access verification by displaying either a correct password indication or an incorrect password indication using LEDs

Working of Password Checker Using Logic Gates



The logic gate based password checker operates entirely through hardware logic using XNOR gates, AND gates, and a NOT gate to verify the entered password. The system consists of three seven-segment displays, each controlled by a 7490 counter IC, allowing the user to increment digits using switches.

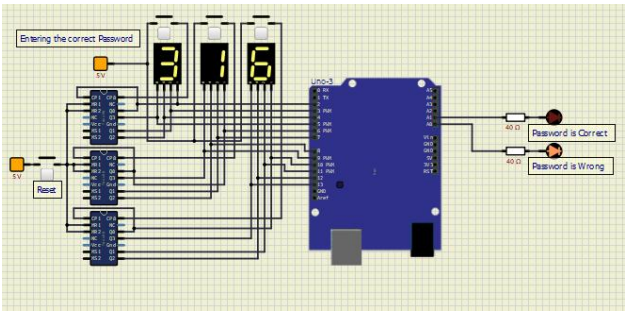
Each seven-segment display represents one digit of the password. As the user presses the switches, the counters increment the displayed values. These output values are continuously compared against a predefined password using XNOR gates. Since XNOR gates produce a HIGH output only when both inputs are identical, each entered digit is checked against the corresponding correct digit.

If all three digits match the predefined password, the outputs of the XNOR gates are combined through AND gates, producing a HIGH signal that activates the correct password LED. If even one digit does not match, the system generates an

incorrect output signal, activating the wrong password LED through a NOT gate arrangement.

Since the password is permanently embedded in the hardware connections, modifying the password requires physical changes to the circuit design and gate connections.

Working of Password Checker Using Arduino UNO



The Arduino-based password checker performs the same operation using software programming instead of dedicated logic gate circuitry. Similar to the logic gate system, the circuit uses three seven-segment displays with 7490 counter ICs, where switches allow users to increment and select password digits.

The Arduino continuously reads the output values generated by the counters and compares them with a predefined password stored in the program code. If all entered digits match the stored password, the Arduino activates the correct password LED. If any digit is incorrect, the wrong password LED is activated.

Unlike the logic gate system, the correct password in the Arduino implementation is not fixed through hardware but is stored inside the program. This allows the password to be easily modified by updating the software code without requiring hardware redesign. As a result, the Arduino approach offers better flexibility and easier maintenance.

Difference Between Logic Gate-Based and Arduino-Based Password Checker

Although both systems perform password verification, their implementation approaches differ significantly.

The logic gate-based password checker relies completely on hardware logic for password comparison. The correct password is directly wired into the circuit through XNOR gate configurations, making the system fast and efficient with very low output delay. However, changing the password requires redesigning the hardware connections, making the system less flexible and difficult to upgrade.

In contrast, the Arduino-based password checker performs password verification through software logic. The correct password is stored in the Arduino program, allowing easy modifications without changing the circuit hardware. This makes the system more scalable and adaptable for future improvements, such as increasing password length or adding additional authentication features.

However, the Arduino system introduces slightly higher output delay because the microcontroller processes inputs through software execution. Additionally, the cost is higher due to the inclusion of the Arduino UNO board.

Therefore, logic gate implementation is more suitable for fixed-password systems requiring fast hardware response, whereas Arduino implementation is better for systems requiring password modification, scalability, and additional programmable features.

Comparative Analysis of Password Checker		
Parameters	**Logic Gates**	**Arduino UNO**
Components Used	12 XNOR gates, 1 NOT gate, 7 AND gates, 2 resistors, 2 LEDs, 3 seven-segment displays, switches, 3 7490 counter ICs	Arduino UNO, 2 resistors, 2 LEDs, 3 seven-segment displays, switches, 3 7490 counter ICs
Scalability	Low (~30%)	High (~80%)
Flexibility	Low (~20%)	High (~80%)
Circuit Complexity	High (~70%)	Moderate (~50%)
Output Delay	~150 ns	~800 ns
Cost Required	₹200–₹300	₹1000–₹1200

Result and Discussion

The comparative analysis of the Password Checker system showed significant differences between hardware-based and software-based implementations. The logic gate-based system demonstrated faster response due to direct hardware-level password comparison through XNOR and AND gate combinations. Since the comparison operation is performed entirely using electronic gates, the output delay remained very low.

However, the hardware implementation required a large number of components, increasing circuit complexity. Additionally, changing the password required modifications to gate connections, reducing system flexibility and scalability.

On the other hand, the Arduino-based system simplified password verification through software programming. The predefined password could be easily changed by modifying the program code without redesigning the circuit. This provided better flexibility and easier expansion possibilities, especially for systems requiring advanced authentication features.

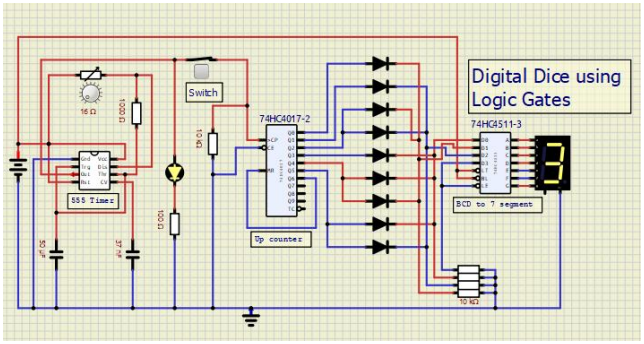
Although the Arduino implementation showed slightly higher output delay and higher cost, its lower hardware complexity and programmable nature made it more practical for modern embedded applications requiring customization and future upgrades.

Based on the comparative study, it can be concluded that both password checker systems effectively verify entered passwords, but each approach offers different advantages. The logic gate-based system provides faster operation and lower cost, making it suitable for fixed-password applications where hardware speed is important. However, its lack of flexibility and higher circuit complexity make modifications difficult.

3. Digital Dice

The objective of the Digital Dice system is to simulate the functioning of a conventional dice using electronic components through both digital logic gate/counter-based implementation and an Arduino UNO microcontroller-based implementation. The circuit is designed to generate continuously changing random digits on a seven-segment display, which stop at a fixed value when the user releases or turns OFF the switch, thereby simulating a dice roll.

Working of Digital Dice Using Logic Gates



The logic gate-based Digital Dice system operates using a 555 Timer IC, 74HC4017 counter IC, and 74HC4511 seven-segment decoder IC**.** The 555 timer is configured to generate continuous clock pulses at high speed, which act as input signals for the counter IC.

As long as the switch remains ON, the timer continuously generates pulses that rapidly increment the counter outputs. Due to the high-speed counting, the seven-segment display continuously changes numbers, creating the

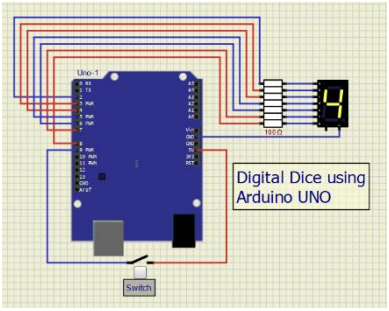
appearance of random digit generation similar to a rolling dice.

The 74HC4017 counter IC advances through count sequences with each timer pulse, while the 74HC4511 decoder IC converts the counter outputs into signals suitable for driving the seven-segment display. Additionally, diodes are used to control and map the counting sequence, ensuring proper digit display.

When the switch is turned OFF, the clock pulses stop, freezing the counter at its current count value. As a result, the seven-segment display stops changing and displays a fixed number, representing the final dice output.

Since the system depends entirely on hardware-level timing and counting operations, it provides fast response and low output delay.

Working of Password Checker Using Arduino UNO



The Arduino-based Digital Dice system performs the same operation using software programming. A switch input is connected to the Arduino UNO, and a seven-segment display is used to display the generated numbers.

When the switch is turned ON, the Arduino continuously executes a programmed randomization process, generating rapidly changing random numbers and displaying them on the seven-segment display. This creates a visual effect similar to a rolling dice.

Once the switch is turned OFF, the Arduino stops the randomization process and retains the currently displayed number as the final dice output. The randomness and display control are entirely managed through software code rather than dedicated hardware timing circuits.

Unlike the logic gate implementation, modifications to the dice behavior—such as changing the speed of randomization, increasing the number range, or adding special effects—can be achieved easily through code modifications without altering the hardware circuit.

Difference Between Logic Gate-Based and Arduino-Based Password Checker

Although both systems successfully simulate the operation of a digital dice, their implementation methods differ significantly.

The logic gate/counter-based system relies on hardware timing and counting mechanisms using a 555 timer and counter ICs. Randomization is achieved through rapid pulse generation and counter increments, resulting in very low output delay and fast hardware response. However, modifying the system or adding additional features requires changes in hardware configuration, reducing flexibility.

In contrast, the Arduino-based system uses software-generated randomization controlled through programming logic. The Arduino code manages the digit generation process, making it easier to modify the dice behavior, customize display patterns, or expand functionality. This provides greater flexibility and scalability.

However, the Arduino system introduces slightly higher output delay due to software execution and increases overall cost because of the microcontroller board. Despite this, the circuit complexity is significantly lower compared to the hardware-based implementation.

Therefore, logic gate implementation is better suited for dedicated low-delay hardware systems, whereas Arduino implementation is more suitable

for programmable and customizable embedded applications.

Comparative Analysis of Digital Dice		
Parameters	**Logic Gates**	**Arduino UNO**
Components Used	555 Timer IC, 74HC4017 IC, 74HC4511 IC, 2 capacitors, 6 resistors, 9 diodes, 1 seven-segment display, switch	Arduino UNO, 7 resistors, 1 seven-segment display, switch
Scalability	Low (~30%)	High (~80%)
Flexibility	Low (~20%)	High (~90%)
Circuit Complexity	High (~70%)	Low (~20%)
Output Delay	~500 ns	~500 ns
Cost Required	₹500–₹600	₹1000–₹1200

Result and Discussion

The comparative analysis of the Digital Dice system highlighted clear differences between hardware-based and software-based implementations. The logic gate/counter-based system achieved faster response and lower output delay due to hardware-driven timing and counting operations using the 555 timer and counter ICs. Additionally, the cost of implementation was comparatively lower because only discrete components and ICs were used.

However, the hardware implementation involved greater circuit complexity due to the inclusion of multiple integrated circuits, resistors, capacitors, and diode arrangements. Modifying system behavior or adding features required hardware redesign, limiting flexibility.

On the other hand, the Arduino-based system significantly reduced hardware complexity by implementing randomization through software programming. The use of code made it easier to modify randomization speed, display behavior, and functionality without changing the physical circuit. This resulted in higher scalability and flexibility for future improvements.

Although the Arduino implementation showed slightly higher output delay and cost, its programmable nature and simplified circuit design made it more suitable for adaptable embedded system applications.

Based on the comparative study, it can be concluded that both implementations successfully perform digital dice simulation, but each offers different advantages. The logic gate/counter-based system provides faster response time and lower implementation cost, making it suitable for dedicated hardware applications requiring minimal delay.

CONCLUSION

This research paper presented a comparative analysis between Arduino microcontroller-based systems and digital logic gate/counter-based systems for small embedded system applications using three practical implementations: Water Level Indicator, Password Checker, and Digital Dice. The study aimed to evaluate the performance and suitability of both approaches based on parameters such as components used, scalability, flexibility, circuit complexity, output delay, and implementation cost.

The findings of the research demonstrated that both Arduino and digital logic gate systems are capable of effectively performing embedded system tasks; however, their suitability varies depending on application requirements. The logic gate/counter-based systems exhibited lower output delay and faster response due to direct hardware-level execution. Additionally, these systems generally required lower implementation cost, making them suitable for fixed-function applications where speed and cost efficiency are important.

On the other hand, the Arduino-based systems showed significant advantages in terms of scalability, flexibility, and ease of modification. Since functionality is controlled through software programming, changes can be implemented without redesigning the hardware circuitry. This makes Arduino more practical for applications requiring

customization, upgrades, or additional functionalities in the future.

In the Water Level Indicator, Arduino simplified implementation and provided higher flexibility, while logic gates delivered faster hardware response. In the Password Checker, Arduino enabled easier password modification through software, whereas logic gates provided dedicated and faster password verification. Similarly, in the Digital Dice, Arduino reduced hardware complexity through code-based randomization, while the logic gate/counter implementation offered faster operation with lower delay.

Overall, the study concludes that digital logic gates are more suitable for simple, fixed, and high-speed applications, whereas Arduino microcontrollers are more advantageous for flexible, scalable, and programmable embedded systems. Therefore, the choice between the two approaches should be made based on the specific requirements of the application, including cost, performance, complexity, and future expandability.

REFERENCES

1. *Ankush and A. S. Bhandari, "Review Paper on Reversible Logic Gate," International Journal of Research in Electrical and Electronics Engineering (IJREEE), vol. 2, no. 7, pp. 01–06, Jul. 2016..*
2. *M. M. Mano and M. D. Ciletti, Digital Design, 5th ed. Pearson Education, 2013.*
3. *Hari Kishan Kondaveeti, Nandeesh Kumar Kumaravelu, Sunny Dayal Vanambathina, Sudha Ellison Mathe, and Suseela Vappangi, "A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations," Computer Science Review, vol. 40, 2021, Art. no. 100364.*
4. *Arduino, "Arduino UNO Rev3 Documentation," Arduino Official Documentation. Available: https://www.arduino.cc/*
5. *P. McRoberts, Beginning Arduino, 2nd ed. Apress, 2013.*
6. *Hudedmani, Mallikarjun, and I. K. Vivek, "Digital Logic Gate Simulation using Arduino Microcontroller," 2017.*
7. *Texas Instruments, NE555 Timer Datasheet.*
8. *Texas Instruments, 7490 Decade Counter IC Datasheet.*
9. *F. Vahid and T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction, Wiley, 2002.*
10. *T. L. Floyd, Digital Fundamentals, 11th ed. Pearson Education, 2015.*