eISSN: 2477-7994

JMECS (Journal of Measurements, Electronics, Communications, and Systems)

Vol.13, No.1, Pg.42-55, June 2026

Goal-Based Autonomous Agent Implementation for an Arduino Uno-Based Embedded Game in a Simulation-Based Environment

Yoseph Satria Praka[1], Ananda Maria[1], Adinda Azhsyara[1], Candi Rahayu Tri Prameswari[1], Desna Romarta Tambun[1], Nurul Wahyuni[1], Dwi Ermawati[2]

[1] *Digital Multimedia Engineering, Jakarta State Polytechnic, Depok, 16425, Indonesia*

[2] *Multimedia and Networking Engineering, Jakarta State Polytechnic, Depok, 16425, Indonesia*

[*]*Corresponding author : yoseph.satria.praka.tik23@stu.pnj.ac.id*

Manuscript received August 14, 2026; revised November 5, 2026; accepted November 9, 2026.

Abstract

The implementation of intelligent agents in embedded game systems provides an opportunity to incorporate autonomous decision-making into microcontroller-based applications. This study presents *Rebutan!*, an Arduino Uno-targeted game featuring an autonomous agent that pursues randomly generated targets during gameplay. The study employed the Four-D model, consisting of Define, Design, Develop, and Disseminate, as the development framework. The prototype was implemented and evaluated as a simulation-based system using the Wokwi environment, incorporating an Arduino Uno, four push buttons, a 32 × 8 MAX7219 LED matrix, an I2C LCD, and a buzzer. The autonomous agent was implemented as a Goal-Based Agent using a Manhattan-Distance-Based Greedy Movement policy with a deterministic axis-priority rule. Black Box Testing evaluated 14 functional scenarios, all of which passed. Gameplay behavior was subsequently evaluated through 50 trials, each lasting 60 seconds, involving five human players with 10 trials per player. The player won 25 trials (50.0%), the AI won 22 trials (44.0%), and 3 trials (6.0%) ended in draws. The mean scores were 11.64 for the player and 11.22 for the AI, with a paired-samples t-test showing no statistically significant difference between the scores, $t(49) = 0.441$, $p = 0.661$. Runtime measurements recorded an estimated peak SRAM usage of 704 bytes (34.38%) and an average AI decision time of 1,698.38 µs within the Wokwi simulation. These findings demonstrate the feasibility of implementing deterministic autonomous target pursuit in a simulation-based embedded game prototype under the defined gameplay and evaluation conditions.

Keywords: 4D Model; Arduino Uno; Embedded Game; Goal-Based Agent; Manhattan Distance.

DOI: 10.25124/jmecs.v13i1.12035

1. Introduction

The advancement of embedded systems has expanded the application of microcontrollers beyond automation and Internet of Things (IoT) systems into interactive digital games and educational platforms. Among various microcontroller platforms, Arduino Uno has become one of the most widely adopted due to its affordability, ease of programming, open-source ecosystem, and compatibility with numerous peripheral devices. These characteristics make Arduino Uno an attractive platform for developing interactive games that combine hardware and software in a compact embedded environment [1], [2]. Nevertheless, most Arduino-based games emphasize user interaction and hardware integration, while the implementation of intelligent autonomous agents capable of making decisions during gameplay remains relatively limited.

Artificial Intelligence (AI) has become an essential component in modern game development because it enables non-player characters (NPCs) or agents to perceive their environment, determine appropriate actions, and pursue predefined objectives autonomously [3], [4]. One of the fundamental applications of AI in grid-based games is pathfinding, which allows an agent to determine an efficient movement path toward a target [5]. Various pathfinding algorithms, such as A*, Dijkstra, Breadth-First Search (BFS), and Greedy Best-First Search,

The online version can be found at https://journals.telkomuniversity.ac.id/jmecs/article/view/12035/3765

DOI: 10.25124/jmecs.v13i1.12035

have been widely implemented in computer games [6], [7], [8].

Several previous studies have demonstrated the potential of Arduino as a platform for interactive game development. A sensor-based controller for the Angry Birds game successfully replaced conventional touchscreen interaction by utilizing physical hand movements, achieving an interaction accuracy of 99.1% while improving user engagement [9]. Another study developed a low-cost interactive hardware game designed to encourage children's physical activity and collaborative play, illustrating the educational potential of Arduino-based gaming systems [10]. Arduino has also been implemented in educational games through the development of a physical *Lights Out!* board to support mathematical modeling and programming instruction, demonstrating its effectiveness in enhancing computational thinking skills [11]. Furthermore, RFID technology integrated with Arduino has been utilized to create an educational card game that introduces marine animals through interactive audio feedback, improving children's learning experiences [12]. Another study introduced Arduino-based LED race and LED matrix games as demonstration media to promote computer science education, emphasizing accessibility and ease of hardware implementation [13].

Although previous studies have demonstrated the feasibility of Arduino for educational and interactive games, their primary contributions have focused on hardware interaction, user interfaces, educational content, and sensor integration. The use of autonomous game agents for real-time decision-making represents a different application focus within Arduino-based game development. In particular, simple coordinate-based movement strategies that do not require explicit path-search structures remain relevant for embedded game applications. This context motivates the exploration of a goal-oriented agent that can perform autonomous target pursuit using coordinate-based calculations within a simulation-based environment.

To address this context, this study implements a goal-based autonomous agent in an Arduino Uno-based game named Rebutan! within a 32 × 8 grid environment displayed using MAX7219 LED matrices. The system is developed and evaluated in a simulation-based environment using Wokwi. The implemented agent uses a Manhattan-Distance-Based Greedy Movement policy for autonomous target pursuit. The agent first selects the nearest active target by calculating the Manhattan Distance between the AI position and each active target. After selecting a target, the agent calculates the horizontal and vertical coordinate differences and determines the next movement using a deterministic axis-priority policy. The horizontal axis is prioritized when the absolute horizontal difference is greater than or equal to the vertical difference; otherwise, the vertical axis is prioritized. When the prioritized axis reaches zero, movement continues along the remaining axis. The approach does not perform explicit graph search, frontier exploration, or path reconstruction and therefore is not a Greedy Best-First Search algorithm. Instead, it implements a deterministic coordinate-based movement policy for autonomous target pursuit in the simulation-based game environment.

The contribution of this research lies in the implementation and evaluation of a goal-based autonomous game agent within an embedded game prototype developed in a simulation-based environment. Unlike Arduino-based game implementations that primarily emphasize interaction, sensing, or educational functionality, this study focuses on autonomous movement decision-making during real-time gameplay. The contribution centers on integrating a deterministic Manhattan-Distance-Based Greedy Movement policy with player input, multiple target generation, score accumulation, and time-limited game-state management in a 32 × 8 embedded game environment. The system's behavior is evaluated through 50 gameplay trials conducted in the Wokwi simulation environment. Thus, the findings provide evidence of the agent's behavior and gameplay feasibility within the simulation-based environment, rather than establishing performance on physical Arduino Uno hardware.

2. Research Method

This study used the Four-D (4D) framework, consisting of Define, Design, Develop, and Disseminate, to organize the development of the *Rebutan!* embedded game and the implementation of its autonomous movement agent. The framework was used as a systematic development structure rather than as a methodological novelty [14], [15], [16]. The research process covered requirement identification, game and agent design, simulation-based implementation, and preparation of implementation documentation. The research process is presented in Figure 1.

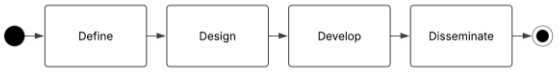

Fig. 1. Research Stage

2.1 Define Stage

The Define stage was conducted to identify the functional requirements and system specifications of the *Rebutan!* Game [17]. The requirements included player movement through four directional buttons, autonomous movement of the AI agent, multiple target generation, score accumulation, time-limited gameplay, visual feedback through a 32 × 8 MAX7219 LED matrix, score and time information

through a 16 × 2 I2C LCD, and audio feedback through a buzzer. The game was specified as a 32 × 8 grid with four-directional movement and a 60-second gameplay duration. The AI was defined as a goal-based autonomous agent that selects the nearest active target using Manhattan Distance and moves toward the selected target according to a deterministic axis-priority policy. The initial positions were defined at the bottom-left corner for the player and the top-right corner for the AI. The system requirements were subsequently used as the basis for the game architecture and agent movement design.

2.2 Design Stage

The Design stage focused on designing the game structure, movement mechanism, target generation, interface, and agent decision procedure based on the requirements identified in the Define stage. The agent first evaluates all active targets and calculates their Manhattan Distance using [18], [19], [20]:

$$D = |x_{ai} - x_t| + |y_{ai} - y_t| \quad (1)$$

The next movement is determined from the horizontal and vertical differences between the AI and the selected target. The horizontal axis is prioritized when $|dx| \geq |dy|$, while the vertical axis is prioritized when $|dx| < |dy|$. The AI performs one-cell movement toward the selected target at each decision interval. The design also specifies five simultaneously active targets and a minimum Manhattan spacing of four units between newly generated targets and existing game entities whenever a valid position can be found.

2.3 Develop Stage

The Develop stage involved implementing the designed game as a simulation-based prototype using the Arduino Uno configuration in the Wokwi environment [21]. The implementation used the LedControl library for the four chained MAX7219 LED matrix modules and the LiquidCrystal_I2C library for the 16 × 2 LCD. Four push buttons were connected to digital pins 2–5, the buzzer to pin 8, and the MAX7219 interface used DIN, CLK, and CS pins 12, 11, and 10, respectively. The game was implemented using a non-blocking timing mechanism based on millis(), with the AI movement interval set to 400 ms, a 60-second gameplay duration, and a 180 ms button debounce interval. Target positions were generated using Arduino's random number generation after initializing the random seed with an analog reading from pin A0.

2.4 Disseminate Stage

The Disseminate stage focused on preparing the implementation information required to reproduce the simulation-based prototype [22]. The dissemination materials included the system configuration, component connections, software libraries, game operation, agent movement procedure, and simulation implementation.

2.5 Experimental Setup

The experimental evaluation was conducted in the Wokwi simulation environment using the completed *Rebutan!* prototype. Each gameplay session lasted 60 seconds, during which the player and AI competed to collect active targets. Five targets were active simultaneously. When either the player or AI reached an active target, the corresponding score increased by one, and the collected target was replaced by a newly generated target. The player started at coordinate (0, 7), while the AI started at (31, 0). The AI evaluated its movement every 400 ms, whereas player movement was controlled through four directional buttons with a 180 ms debounce interval. Target generation randomly selected grid coordinates while attempting to maintain a minimum Manhattan spacing of four units from the player, AI, and other active targets. If a valid position could not be obtained after more than 60 attempts, the spacing constraint was relaxed to prevent the game from becoming stalled.

The gameplay evaluation consisted of 50 trials involving five players, with each player completing 10 trials. Each trial started as a new game session, with the player and AI initialized at their predefined starting positions, scores reset to zero, and five new target points generated. The players controlled their characters using the four directional buttons throughout the 60-second session, while the AI operated autonomously according to the implemented movement policy. The final player and AI scores were recorded at the end of each trial. The winner was determined by comparing the accumulated scores, with a higher player score classified as a player win, a higher AI score classified as an AI win, and equal scores classified as a draw. Descriptive statistics were calculated from the 50 trials, including the mean and standard deviation of the player and AI scores, as well as the frequency and percentage of each outcome. A paired-samples t-test was also conducted to examine the difference between player and AI scores across the trials.

The random target positions were not controlled using a fixed random seed because the implementation initializes the random number generator using an analog reading from pin A0. Consequently, target configurations could vary between game sessions. The use of five players and repeated trials was intended to provide observations from multiple human participants rather than relying on a single player's performance.

3. Results and Discussion

3.1 Game Design and System Specification

The *Rebutan!* game was designed as a time-limited competitive embedded game in which a human player and an autonomous agent compete to collect targets on a 32 × 8 grid. The player moves using four directional push buttons, while the agent selects the nearest active target using Manhattan Distance and determines its movement through a deterministic axis-priority policy. The game is implemented as a Wokwi simulation prototype using an Arduino Uno, four chained MAX7219 LED matrix modules, a 16 × 2 I2C LCD, four push buttons, and a buzzer. The main game specifications are presented in Table 1.

Table 1. Game Design and System Specification

Component	Description
Game Title	*Rebutan!*
Genre	Grid-based competitive embedded game
Platform	Arduino Uno using Wokwi simulation
Display	Four chained MAX7219 LED matrix modules forming a 32 × 8 grid and a 16 × 2 I2C LCD
Player Control	Four push buttons for Up, Down, Left, and Right movement
Autonomous Agent	Goal-based agent using a Manhattan-Distance-Based Greedy Movement policy
Environment	32 × 8 grid with four-directional movement
Game Duration	60 seconds per trial
Objective	Collect as many targets as possible within 60 seconds
Active Targets	Five targets are maintained simultaneously
Target Generation	Random grid positions with a minimum Manhattan spacing of four units when a valid position is available
Target Selection	The nearest active target is selected based on Manhattan Distance
AI Decision	Horizontal movement is prioritized when (
AI Movement Interval	400 ms using millis()
Player Input	Four-button input with 180 ms debounce handling
Scoring	One point is awarded for each target collected
Win Condition	Player score is higher than AI score when 60 seconds ends
Lose Condition	AI score is higher than player score when 60 seconds ends
Draw Condition	Player and AI obtain equal scores when 60 seconds ends
Audio Feedback	Buzzer provides movement and game-result feedback

3.2 AI Movement Flow

The AI movement flow describes the decision process used by the autonomous agent to pursue active targets during gameplay. The flow is designed to determine the target to pursue and the movement direction based on the agent's current position and the selected target. The movement process of the agent is illustrated in Figure 2.

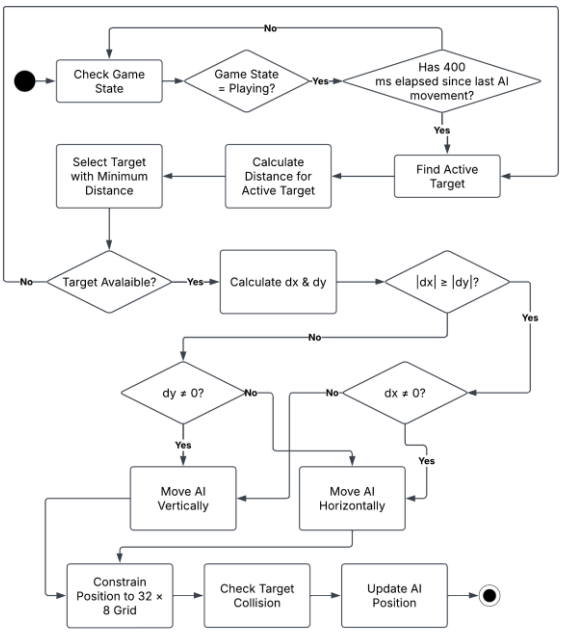

Fig. 2. AI Movement Flow

As shown in Figure 2, the agent first checks whether the game is in the *PLAYING* state and whether 400 ms have elapsed since its previous movement. If both conditions are satisfied, the Manhattan Distance is calculated for each active target, and the target with the smallest distance is

selected. The agent then calculates dx and dy between its position and the selected target. When $|dx| \geq |dy|$ horizontal movement is prioritized, while vertical movement is prioritized otherwise. If the prioritized direction has no remaining distance, the agent moves along the other axis. The resulting position is constrained within the 32 × 8 grid, followed by a collision check with the target before the movement cycle returns to the main game loop.

3.3 Agent Model

The agent in *Rebutan!* is designed as a goal-based agent that continuously determines its movement toward an active target during gameplay [23]. The agent receives the current game state, including its position and the positions of active targets, then calculates the Manhattan Distance to identify the nearest target. Based on the selected target, the agent determines its movement direction using an axis-priority rule and updates its position after movement. The overall agent model is illustrated in Figure 3.

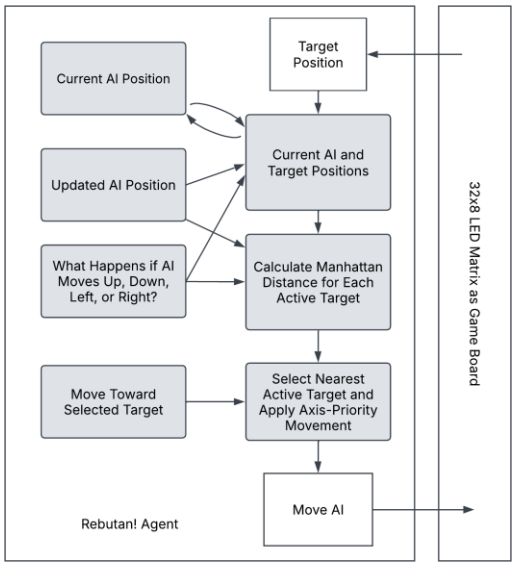

Fig. 3. Agent Model of "Rebutan!"

Figure 3 presents the relationship between the agent's state, target selection, movement decision, and position update within the 32 × 8 game environment. The current AI position and active target positions are provided as the basis for calculating Manhattan Distance. The nearest active target is then selected, after which the agent moves toward the selected target according to the implemented movement policy. Following movement, the updated AI position is returned to the game environment, allowing the process to continue during gameplay.

3.4 System Implementation and Simulation Setup

The system was implemented and tested as a simulation-based prototype using Wokwi with an Arduino Uno as the main controller. The simulation integrates four push buttons for player input, four chained MAX7219 modules forming a 32 × 8 LED matrix as the game arena, a 16 × 2 I2C LCD for game status information, and a buzzer for audio feedback. The Arduino Uno coordinates player input, autonomous agent movement, target generation, score calculation, display updates, and game-state transitions. The component configuration and pin assignments used in the simulation are presented in Table 2. The implemented configuration follows the program structure and hardware definitions used in the prototype.

Table 2. Components and Pin Configuration

Component	Qty	Arduino Pin	Function
Arduino Uno	1	-	Main controller of the embedded game
Push Button (Up)	1	D2	Moves the player upward
Push Button (Down)	1	D3	Moves the player downward
Push Button (Left)	1	D4	Moves the player to the left
Push Button (Right)	1	D5	Moves the player to the right
MAX7219 LED Matrix	4	DIN D12, CLK D11, CS D10	Displays the 32 × 8 game arena, player, AI, and targets
LCD 16 × 2 I2C	1	SDA A4, SCL A5	Displays game status and results
Buzzer	1	D8	Provides audio feedback
5V Power	-	5V	Supplies power to the modules
Ground	-	GND	Common ground connection

The software implementation consists of initialization, player input, autonomous agent movement, target generation, display rendering, collision and scoring, and game-state management functions. During initialization, the Arduino Uno, input and output peripherals, and game variables are

configured, while the player and AI are assigned their initial positions at (0,7) and (31,0), respectively. The program is implemented using Arduino C/C++ with the LedControl and LiquidCrystal_I2C libraries to control the MAX7219 LED matrix and I2C LCD. The player input is handled through four directional buttons with a 180 ms debounce interval, while the autonomous agent is updated at a 400 ms interval using millis(). The game operates for 60 seconds, and five targets are maintained during gameplay. When a target is collected, the corresponding score is increased and the target is respawned.

The target generation process randomly determines target positions while applying a minimum Manhattan spacing of four units from the player, AI, and other active targets when a valid position is available. The program attempts to find a valid position and relaxes the constraint after more than 60 unsuccessful attempts to avoid indefinite generation. The LED matrix is continuously updated to visualize the player, AI, and active targets, while the LCD provides game-state information. At the end of the 60-second session, the final scores are compared to determine a player win, AI win, or draw.

The Wokwi simulation circuit is presented in Figure 4. The configuration represents a simulation-based implementation of the proposed embedded game, consisting of the Arduino Uno, four push buttons, four chained MAX7219 modules forming the 32 × 8 LED matrix, 16 × 2 I2C LCD, and buzzer. The simulation was used to verify the functional and behavioral operation of the implemented game and autonomous agent. It does not represent a physical hardware deployment; therefore, the results obtained from this setup are limited to simulation-based functional feasibility and do not establish physical hardware performance such as actual processing latency, memory utilization, or electrical behavior.

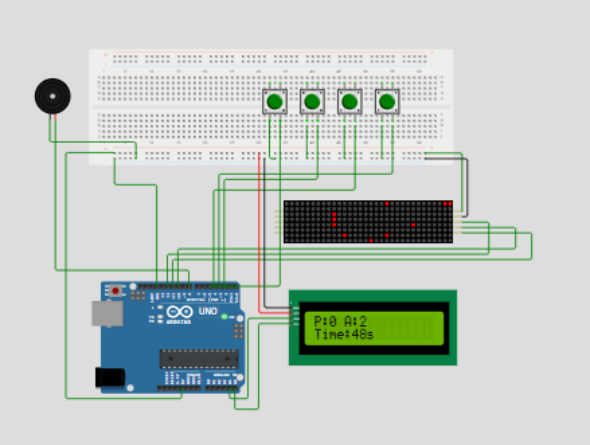

Fig. 4. Wokwi Simulation Circuit of "Rebutan!"

3.5 Testing and Evaluation

Black Box Testing was conducted to evaluate the functional performance of the proposed Rebutan! game without considering the internal implementation of the program code. The testing process focused on verifying whether each input produced the expected output according to the system requirements. All primary functions, including player movement, autonomous AI navigation, point generation, score calculation, timer execution, and game state transitions, were evaluated through a series of functional test scenarios [24], [25]. The complete functional testing results are presented in Table 3.

Table 3. Black Box Testing Result of "Rebutan!"

No.	Test Scenario	Test Data	Expected Result	Actual Result	Status
1	System initialization	Start program	Game initializes successfully and displays the initial game state, including player, AI, target, scores, and timer	Game started successfully with initial player and AI positions, active targets, reset scores, and a 60-second game timer	Pass
2	Player movement	Press Up, Down, Left, and Right buttons	Player moves according to the corresponding button input	Player position changed correctly according to the button input	Pass
3	Player boundary validation	Move player toward arena boundary	Player cannot move outside the 32 × 8 game arena	Player position remained within the 32 × 8 arena limits	Pass

No.	Test Scenario	Test Data	Expected Result	Actual Result	Status
4	AI automatic movement	Wait for the AI movement interval	AI moves automatically according to the predefined movement policy without user interaction	AI position was updated automatically at the predefined movement interval	Pass
5	Autonomous agent decision	Active target positions and AI position	AI selects the nearest active target based on Manhattan Distance and moves toward the selected target according to the predefined axis-priority rule	AI selected the nearest active target and moved toward it according to the predefined movement policy	Pass
6	Point collection by AI	AI reaches an active target	AI score increases by one and a replacement target is generated according to the configured target-generation conditions	AI score increased by one, and a replacement target was generated after the AI collected the target	Pass
7	Point collection by player	Player reaches an active target	Player score increases by one and a replacement target is generated according to the configured target-generation conditions	Player score increased by one, and a replacement target was generated after the player collected the target	Pass
8	Random target generation	Point successfully collected	A replacement target is generated at a random grid position while attempting to maintain a minimum Manhattan spacing of four units from the player, AI, and other active targets	A replacement target was generated after point collection, with the configured spacing condition applied during target generation	Pass
9	Score update	Player or AI collects a target	The corresponding player's score increases by one and the updated score is displayed	Player and AI scores were updated correctly according to the collected targets	Pass
10	Countdown timer	Run game for 60 seconds	Timer decreases continuously from 60 seconds until reaching 0 seconds	Countdown operated normally from 60 seconds to 0 seconds	Pass
11	Game termination and result determination	Timer reaches 0 seconds	Game ends and the final scores are displayed; the side with the higher score is declared the winner, while equal scores result in a draw	Final scores were displayed after the timer reached 0 seconds, and the game result was determined according to the final scores	Pass
12	Automatic restart	Previous game finished	A new game starts automatically with scores reset, five active targets generated, and a 60-second game timer	The system automatically started a new game session with reset scores, newly generated target positions, and a 60-second timer	Pass

No.	Test Scenario	Test Data	Expected Result	Actual Result	Status
13	Continuous execution	Multiple consecutive game sessions	System continues operating across multiple 60-second sessions without functional errors	The system continued to execute consecutive game sessions without observed functional errors	Pass
14	AI boundary validation	AI moves toward the arena boundary	AI remains within the 32 × 8 game boundaries	AI position remained within the 32 × 8 game boundaries	Pass

The results presented in Table 3 indicate that all tested functional requirements were successfully implemented and operated according to their specified requirements. All 14 Black Box Testing scenarios achieved the expected outputs, resulting in a 100% functional success rate. This finding confirms the functional correctness of the implemented game features, including input processing, movement, target generation, scoring, and game-state handling. However, the Black Box Testing results only demonstrate that the system functions according to its defined requirements and do not indicate the effectiveness, efficiency, or optimality of the intelligent agent's movement strategy. To further evaluate the gameplay performance of the proposed Goal-Based Agent, 50 gameplay trials were conducted with five human players, with each player completing 10 trials. The trials were organized into five groups, where Trials 1–10 were conducted by Player 1, Trials 11–20 by Player 2, Trials 21–30 by Player 3, Trials 31–40 by Player 4, and Trials 41–50 by Player 5. Each trial lasted 60 seconds. For every trial, the final scores obtained by the player and the autonomous agent were recorded at the end of the gameplay period. The winner was determined by comparing the final scores, where the higher score indicated a win and an equal score indicated a draw. The results of all 50 trials are presented in Table 4 and Table 5.

Table 4. Gameplay Performance Results

Trial	Player Score	AI Score	Result
1	19	8	Player
2	22	10	Player
3	24	9	Player
4	23	7	Player
5	5	11	AI
6	5	16	AI
7	7	12	AI
8	7	12	AI
9	8	13	AI
10	8	16	AI
11	9	12	AI
12	9	12	AI
13	4	4	Draw
14	9	5	Player
15	10	9	Player
16	10	7	Player
17	12	5	Player
18	14	10	Player
19	18	7	Player
20	14	5	Player
21	4	12	AI
22	7	12	AI
23	8	16	AI
24	5	11	AI
25	4	13	AI
26	6	11	AI
27	6	14	AI
28	18	7	Player
29	4	17	AI
30	8	11	AI
31	6	11	AI
32	11	13	AI
33	10	15	AI
34	14	10	Player
35	13	11	Player
36	15	11	Player
37	12	13	AI
38	14	12	Player
39	13	12	Player
40	14	14	Draw
41	17	12	Player
42	16	16	Draw
43	17	14	Player
44	12	13	AI
45	16	15	Player
46	15	11	Player
47	14	11	Player
48	16	13	Player
49	14	10	Player
50	16	10	Player

Table 5. Summary of the 50 Trials

Metric	Player	AI
Number of Wins	25	22
Win Rate	50.0%	44.0%
Mean Score	11.64	11.22

Metric	Player	AI
Standard Deviation	5.21	3.09
Minimum Score	4	4
Maximum Score	24	17
Total Score	582	561
95% Bootstrap CI of Mean	10.18–13.10	10.36–12.02

The results presented in Table 5 show 50.0% player wins, 44.0% AI wins, and 6.0% draws across the 50 gameplay trials conducted by five human players. The player achieved a slightly higher mean score of 11.64 ± 5.21, compared with 11.22 ± 3.09 for the AI. The corresponding 95% bootstrap confidence intervals for the mean scores were 10.18–13.10 for the player and 10.36–12.02 for the AI. These results show that both the human players and the autonomous agent were able to collect targets and accumulate scores under the defined 60-second gameplay conditions. The experiment does not support a claim that the proposed agent consistently outperformed the human players. Instead, the results demonstrate that the Manhattan-Distance-Based Greedy Movement Agent was able to operate autonomously and participate in the gameplay competition under the defined conditions.

To further examine the difference between player and AI scores, a paired-samples t-test was conducted across the 50 gameplay trials. The mean difference between the player and AI scores was 0.42 points, with a standard deviation of 6.73. The analysis produced a *t*-value of 0.441 with 49 degrees of freedom and a significance value of 0.661. The 95% confidence interval for the mean difference ranged from −1.493 to 2.333. Since the confidence interval includes zero and the *p*-value exceeds 0.05, the observed difference in scores was not statistically significant. Thus, although the player obtained a slightly higher mean score than the AI, the paired comparison did not provide statistical evidence of a difference in mean scores under the tested gameplay conditions.

The descriptive results varied across the five player groups. For Player 1, the player obtained a mean score of 12.80 ± 8.08, compared with 11.40 ± 3.06 for the AI, while Player 2 obtained 10.90 ± 3.81 compared with 7.60 ± 2.99 for the AI. For Player 3, the player obtained a mean score of 7.00 ± 4.16, whereas the AI obtained 12.40 ± 2.84. For Player 4, both the player and AI obtained an identical mean score of 12.20, with standard deviations of 2.66 and 1.55, respectively. For Player 5, the player obtained a mean score of 15.30 ± 1.57, compared with 12.50 ± 2.07 for the AI. These results show that the relative score patterns differed across the five players, with the player mean exceeding the AI mean for Players 1, 2, and 5, while the AI mean exceeded the player mean for Player 3 and both means were equal for Player 4.

The variation in scores also differed among the player groups. Player 1 showed the largest variability in player scores, with a standard deviation of 8.08 and a range of 5–24, whereas Player 5 showed substantially lower variation, with a standard deviation of 1.57 and a range of 12–17. The AI scores generally showed lower variation within each group, with standard deviations ranging from 1.55 for Player 4 to 3.06 for Player 1. The group-level results indicate that gameplay outcomes varied across participants, which should be considered when interpreting the aggregated results from the 50 trials. These descriptive statistics provide information about score variation across participants, while the paired-samples t-test provides the statistical comparison of player and AI scores across the complete set of trials.

The runtime performance of the *Rebutan!* prototype was evaluated through measurements collected during execution in the Wokwi simulation environment. The evaluation focused on SRAM utilization, AI decision execution time, AI movement interval, and main-loop execution time. A sample of the measured results is presented in Table 6.

Table 6. Runtime Performance Measurements

Metric	Result
Initial Free SRAM	1,350 bytes
Minimum Free SRAM	1,344 bytes
Estimated Peak SRAM Usage	704 bytes
SRAM Utilization	34.38%
Average AI Decision Time	1,698.38 µs
Maximum AI Decision Time	49,436 µs
Configured AI Movement Interval	400 ms
Average Measured AI Movement Interval	623.59 ms
Maximum Measured AI Movement Interval	701 ms
Main Loop Iterations	104,065
Average Main Loop Execution Time	578.16 µs
Maximum Main Loop Execution Time	403,152 µs

As presented in Table 6, the prototype recorded an initial free SRAM of 1,350 bytes and a minimum free SRAM of 1,344 bytes during execution. Based on the 2,048-byte SRAM capacity used for the Arduino Uno simulation, the estimated peak SRAM usage was 704 bytes, corresponding to 34.38% utilization. The AI recorded 96 movement decisions during the measured execution, with an average decision time of 1,698.38 µs. The configured AI movement interval was 400 ms, while the measured average interval was 623.59 ms and the maximum interval was 701 ms. The observed interval therefore represents the actual timing

measured during program execution rather than the configured movement interval alone.

The main loop was executed 104,065 times, with an average execution time of 578.16 µs. The maximum AI decision time was 49,436 µs, while the maximum main-loop execution time reached 403,152 µs. These maximum values represent peak observations during the simulation and should not be interpreted as typical execution times. In particular, individual execution intervals can be affected by additional program operations outside the normal AI movement calculation. Overall, the measurements provide runtime evidence of SRAM utilization and execution timing for the implemented prototype in Wokwi. However, these results remain limited to the simulation environment and do not establish physical Arduino Uno performance, including hardware-level execution timing, electrical behavior, or resource utilization on an actual device.

3.6 Dissemination

The dissemination stage of the 4D model was carried out by publishing the developed prototype as an online demonstration that can be accessed through s.pnj.ac.id/Rebutan. This deployment allows users to observe the overall game concept, hardware configuration, intelligent agent implementation, and system workflow before performing physical assembly on the Arduino Uno platform. Therefore, the published prototype serves as both a validation medium and a practical assembly guide, enabling future developers and learners to reproduce and further develop the proposed Rebutan! game with a Goal-Based Agent on embedded systems.

3.7 Discussion

The 50-trial results indicate that the Manhattan-Distance-Based Greedy Movement Agent was able to perform autonomous target pursuit under the defined gameplay conditions. The player obtained a slightly higher mean score than the AI (11.64 versus 11.22), with win rates of 50.0% and 44.0%, respectively, while 6.0% of the trials ended in a draw. The paired-samples t-test showed that the difference between player and AI scores was not statistically significant, $t(49) = 0.441$, $p = 0.661$, with a 95% confidence interval of −1.493 to 2.333. The AI also showed lower score variation than the player, with standard deviations of 3.09 and 5.21, respectively. At the participant level, the relative results varied across the five players, indicating that gameplay outcomes were influenced by differences among participants. These findings indicate that the agent could operate autonomously and compete with human players under the defined conditions, while the observed results do not provide statistical evidence of a difference in mean scores or evidence of algorithmic superiority.

Compared with previous studies, this work introduces a different contribution by embedding artificial intelligence directly into an Arduino-based game. Previous research has explored sensor-based game controllers [9], low-cost physical games for promoting active play [10], Arduino games for educational purposes [11], RFID-based interactive learning games [12], and LED matrix games for technology demonstrations [13]. Within this context, the contribution of the present study is not the introduction of a new pathfinding algorithm, but the implementation of a Manhattan-Distance-Based Greedy Movement Agent as a simple goal-oriented decision mechanism within an embedded game environment. The repeated 50-trial evaluation provides behavioral evidence that the agent can autonomously operate across multiple 60-second gameplay sessions. However, these results should be interpreted as evidence of functional and behavioral feasibility under the tested conditions rather than as evidence of algorithmic superiority or computational efficiency.

Several limitations restrict the generalization of these findings. First, the experiments were conducted entirely in the Wokwi simulation environment, so the results establish simulation-based functional and behavioral feasibility rather than physical Arduino Uno performance. Second, the experiment involved five human players with 10 trials per player, and the observed differences among player groups indicate that participant-related variation may influence the aggregated results. Third, the current experiment did not explicitly control or report the initial distance between each agent and the selected target, while target positions were generated dynamically during gameplay. These factors may contribute to variation in final scores. Fourth, no baseline comparison with random movement, BFS, A*, or true Greedy Best-First Search was performed. Future studies should control or systematically vary initial conditions and target-generation parameters, compare alternative agent strategies under equivalent conditions, and introduce obstacle-based environments to examine how the movement policy behaves when direct movement toward a target is no longer always possible. Physical Arduino Uno testing would also be necessary to evaluate processing latency, memory utilization, and other hardware-level characteristics that cannot be established from the Wokwi simulation.

4. Conclusions

This study demonstrates the simulation-based feasibility of implementing a Goal-Based Agent through a Manhattan-Distance-Based Greedy Movement policy in the *Rebutan!* game using the Wokwi environment. The implemented agent autonomously selected active targets based on Manhattan Distance and determined its movement

using a deterministic axis-priority policy within a 32 × 8 grid. Across 50 gameplay trials conducted by five human players, the player obtained a 50.0% win rate, while the AI obtained 44.0%, with 6.0% of trials resulting in draws. The mean scores were 11.64 for the player and 11.22 for the AI. A paired-samples t-test showed that the difference between player and AI scores was not statistically significant, $t(49) = 0.441$, $p = 0.661$, with a 95% confidence interval of −1.493 to 2.333. These findings indicate that the implemented deterministic movement policy enabled the agent to autonomously participate in the gameplay under the defined 60-second conditions, without providing statistical evidence of a difference in mean scores between the player and AI. The main contribution of this study is the implementation and repeated behavioral evaluation of a goal-oriented autonomous movement mechanism based on coordinate comparison and Manhattan Distance within an Arduino Uno-targeted embedded game.

The findings should be interpreted within the boundaries of the Wokwi simulation environment and the tested gameplay conditions rather than as evidence of physical hardware performance or algorithmic superiority. The evaluation was limited to five human players, randomly generated target configurations, and an obstacle-free 32 × 8 environment, which may influence the observed gameplay outcomes and limits the applicability of the movement policy to more complex navigation scenarios. Runtime measurements provided additional information regarding SRAM utilization and execution timing within the simulation, but these measurements do not establish the corresponding characteristics of a physical Arduino Uno. Future work should validate the system on physical Arduino Uno hardware, introduce obstacle-aware environments, compare the proposed movement policy with alternative autonomous and pathfinding strategies, investigate dynamic target-selection mechanisms, systematically control initial conditions and target-generation parameters, and further evaluate computational characteristics such as memory utilization, processing time, and decision latency under physical hardware conditions.

References

[1] M. I. Rosli, S. Omar, A. N. Jaafar, R. Abdullah, N. Fadzal, dan M. W. Rosli, "Implementing Arduino in the Classic Gameplay Marble Maze Labyrinth," in *2024 IEEE International Conference on Applied Electronics and Engineering (ICAEE)*, IEEE, Jul 2024, hal. 1–6. doi: 10.1109/ICAEE62924.2024.10667621.

[2] V. Shevchuk, O. Stepanov, H. Klym, T. Tkachuk, dan M. Khvyshchun, "SPECIALIZED GAME COMPUTER SYSTEM BASED ON ARDUINO UNO," *Electron. Inf. Technol.*, vol. 24, hal. 23–33, 2023, doi: 10.30970/eli.24.3.

[3] H. Armanto, H. A. Rosyid, Muladi, dan Gunawan, "Improved Non-Player Character (NPC) behavior using evolutionary algorithm—A systematic review," *Entertain. Comput.*, vol. 52, hal. 100875, Jan 2025, doi: 10.1016/j.entcom.2024.100875.

[4] Y. Karaca, D. Derias, dan G. Sarsar, "AI-Powered Procedural Content Generation: Enhancing NPC Behaviour for an Immersive Gaming Experience," *SSRN Electron. J.*, 2024, doi: 10.2139/ssrn.4663382.

[5] R. Tolentino, "A*Grid2D: Optimized Grid-Based A* Pathfinding for 2D Game Environments," *Business, Educ. Soc. Sci. Technol.*, vol. 1, no. 2, hal. 219–224, Jul 2025, doi: 10.69478/BEST2025v1n2a033.

[6] S. L. Pardede, F. R. Athallah, Y. N. Huda, dan F. D. Zain, "A Review of Pathfinding in Game Development," *[CEPAT] J. Comput. Eng. Progress, Appl. Technol.*, vol. 1, no. 01, hal. 47–56, Mei 2022, doi: 10.25124/cepat.v1i01.4863.

[7] S. R. Lawande, G. Jasmine, J. Anbarasi, dan L. I. Izhar, "A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games," *Appl. Sci.*, vol. 12, no. 11, hal. 5499, Mei 2022, doi: 10.3390/app12115499.

[8] A. Elshahed, M. K. Bin Majahar Ali, A. S. A. Mohamed, F. A. B. Abdullah, dan T. L. J. Aun, "Efficient Pathfinding on Grid Maps: Comparative Analysis of Classical Algorithms and Incremental Line Search," *IEEE Access*, vol. 13, hal. 98473–98484, 2025, doi: 10.1109/ACCESS.2025.3575168.

[9] M. Zinaty, E. El Ghaoui, R. A. Zeid Daou, A. Hayek, dan J. Boercsoek, "Design and Implementation of an Angry Birds Game Controller Using Flex Sensor and a Potentiometer," in *2023 IEEE 4th International Multidisciplinary Conference on Engineering Technology (IMCET)*, IEEE, Des 2023, hal. 1–6. doi: 10.1109/IMCET59736.2023.10368237.

[10] V. Sharma, R. Tiwari, R. Bhargava, R. Jakhar, dan S. R. Rituraj, "A design study of embedded system for a hardware assembly of snake and ladder game," in *RACON*, 2023, hal. 1–3. [Daring]. Tersedia pada: https://pratibodh.org/index.php/pratibodh/article/view/73

[11] J. Gak, V. Artigue, dan J.-J. Flores-Godoy, "Innovation in Linear Algebra Teaching:

Modelling and Solving the 'Lights Out!' Game with Python and Arduino," in *2026 Argentine Conference on Electronics (CAE)*, IEEE, Mar 2026, hal. 67–70. doi: 10.1109/CAE69023.2026.11450339.

[12] L. Wibowo dan R. Wahyusari, "IMPLEMENTASI ARDUINO DAN KARTU RFID PADA MEDIA PEMBELAJARAN MENGENAL HEWAN LAUT," *J. Inform. dan Rekayasa Elektron.*, vol. 5, no. 2, hal. 253–262, Nov 2022, doi: 10.36595/jire.v5i2.676.

[13] M. F. Pereira dan M. S. Oyamada, "Applying Arduino and individually addressable LED strips to the development of interactive materials to promote computer science undergraduate course," in *Anais do XX Congresso Latino-Americano de Software Livre e Tecnologias Abertas (Latinoware 2023)*, Sociedade Brasileira de Computação - SBC, Okt 2023, hal. 97–101. doi: 10.5753/latinoware.2023.236525.

[14] Suprianto, F. Rofah, E. I. Kamariyah, dan M. Sumo, "Development of Physics Providers Based on Arduino Uno Assisted With Infrared Sensors on Viscosity Materials," *SEJ (Science Educ. Journal)*, vol. 9, no. 1, hal. 1–21, Mei 2025, doi: 10.21070/sej.v9i1.1676.

[15] Jufrida, H. S. Falah, dan N. H. Sehab, "Development of An Audio Frequency Meter with Arduino and MAX4466 Sound Sensor," *J. Penelit. Pendidik. IPA*, vol. 9, no. 11, hal. 10280–10286, Nov 2023, doi: 10.29303/jppipa.v9i11.6144.

[16] M. A. Firdaus, S. Subhan, N. D. Arisandi, Z. Zulkarnain, H. Hamzah, dan J. Sabaryati, "Development of Arduino Uno-Based Free Fall Motion Props to Increase Learning Motivation Towards Practicum Results of 11th Grade Students," *J. Educ. FKIP UNMA*, vol. 10, no. 2, hal. 520–529, Jun 2024, doi: 10.31949/educatio.v10i2.8494.

[17] E. W. Hardianti, "Analysis of the Needs for Development of Kahoot Game-Based Learning Media," *IJOEM Indones. J. E-learning Multimed.*, vol. 2, no. 2, hal. 66–72, Mei 2023, doi: 10.58723/ijoem.v2i2.187.

[18] K. Gibran, R. Heriansyah, dan Z. Romegar Mair, "Penerapan Algoritma A-Star sebagai Metode Pathfinding pada Game Lokasi Logika Cerdas untuk Non-Player Character," *Insect (Informatics Secur. J. Tek. Inform.*, vol. 12, no. 01, hal. 158–167, Mar 2026, doi: 10.33506/insect.v12i01.5508.

[19] M. Mihuandayani, D. Mokodompit, dan M. Tomatala, "Case-Based Reasoning for Elderly Eye Disease Diagnosis Using Manhattan Distance and Dempster-Shafer," *CogITo Smart J.*, vol. 12, no. 1, hal. 43–56, Jun 2026, doi: 10.31154/cogito.v12i1.1050.43-56.

[20] A. Z. Putra, S. Yek, S. C. Kwok, E. Tarigan, dan W. F. Sego, "Fingerprint Identification for Attendance Using Euclidean Distance and Manhattan Distance," *sinkron*, vol. 7, no. 4, hal. 2345–2352, Okt 2023, doi: 10.33395/sinkron.v8i4.12844.

[21] D. Darmaji, M. Jamlan, R. Yildiz, A. K. Boglou, dan Z. H. Z. Baharin, "Utilization of Wokwi Technology as a Modern Electronics Learning Media," *J. Educ. Technol. Learn. Creat.*, vol. 2, no. 2, hal. 256–268, Des 2024, doi: 10.37251/jetlc.v2i2.1392.

[22] A. Gledis H, Y. A. Djawad, dan H. Jaya, "PENGEMBANGAN MEDIA TRAINER ROBOTIKA BERBASIS INTERNET OF THINGS DI SEKOLAH MENENGAH KEJURUAN," *UNM J. Technol. Vocat.*, vol. 9, no. 2, hal. 117–127, 2025, doi: https://doi.org/10.26858/ujtv.v9i2.6494.

[23] Y. S. Praka, C. R. T. Prameswari, N. Wahyuni, dan A. Maria, "Implementation of a Pathfinding-Based Intelligent Agent for Enemy Behavior in the 'Run to BIPOL' Maze Tournament Game," *J. Games Game Art Gamification*, vol. 11, no. 2, hal. 38–45, Agu 2026, doi: 10.21512/jggag.v11i2.16155.

[24] A. Anisa, S. Paembonan, R. Suppa, M. Muhallim, Mukramin, dan B. Sulaeman, "RANCANG BANGUN ALAT PENGUKUR TINGGI BADAN OTOMATIS MENGGUNAKAN ARDUINO PADA POSYANDU KELURAHAN SAMPODDO," *J. Inform. dan Tek. Elektro Terap.*, vol. 13, no. 1, hal. 986–996, Jan 2025, doi: 10.23960/jitet.v13i1.5792.

[25] I. Sonjaya, N. Marcheta, dan M. Huzaifa, "Pemanfaatan Teknologi Virtual Reality Dengan Model Game First Person Shooter (FPS) Untuk Pengembangan Laboratorium Virtual Teknologi Audio," in *Seminar Nasional Inovasi Vokasi*, 2023, hal. 153–157. [Daring]. Tersedia pada: https://prosiding.pnj.ac.id/sniv/article/view/416

Author information



Yoseph Satria Praka is a student at Politeknik Negeri Jakarta, Indonesia. He is interested in information technology, educational technology, and *software* development. Throughout his academic journey, he has participated in various projects and activities related to digital innovation and learning technologies. His current work focuses on the application of technology to support education and improve learning experiences.



Ananda Maria is a Digital Multimedia Engineering student at Jakarta State Polytechnic with a strong interest in digital media and creative design. Throughout her studies, she has developed skills in visual design, video editing, and multimedia content production through various academic projects. She believes that creativity and technology can be combined to create informative and engaging works. With these skills, she strives to develop digital content that effectively communicates messages and provides meaningful experiences for audiences.



Adinda Azhsyara is a Digital Multimedia Engineering student at Politeknik Negeri Jakarta with a strong interest in interactive multimedia and game development. She is interested in understanding how digital products are designed and built, from user flow and interface design to the programming process that makes them work effectively.



Candi Rahayu Tri Prameswari is a student at Politeknik Negeri Jakarta with an interest in digital technology and interactive multimedia. Throughout her studies, she has actively developed her knowledge and skills through various projects related to application development and digital media. Her interests include user experience design, digital innovation, and the use of technology to support effective information delivery. She continuously seeks to expand her competencies and keep up with technological advancements to create creative and meaningful solutions.



Desna Romarta Tambun is a student at Politeknik Negeri Jakarta, Indonesia. She is interested in information technology, digital innovation, and software development. Throughout her academic journey, she has been involved in various organizational activities, event committees, and technology-based projects that have helped develop her leadership, communication, and teamwork skills. Her current interests focus on creating digital solutions and educational technologies that can provide positive impacts and meaningful user experiences.



Nurul Wahyuni is a student at Jakarta State Polytechnic, Indonesia. Her academic interests include animation, motion graphics, interactive multimedia, and digital media development. Throughout her studies, she has actively participated in various projects that integrate visual creativity with information technology. Her current work focuses on the use of visual media and digital technologies to create educational, communicative, and engaging content that supports more effective information delivery.



Dwi Ermawati, M.T., is a lecturer in the Applied Bachelor Program of Multimedia and Network Engineering at Politeknik Negeri Jakarta, Indonesia. She has been actively involved in teaching, research, and community engagement activities in the fields of information and communication technology. Her areas of expertise include Internet of Things (IoT), image processing, device control systems, embedded systems, computer networks and security, and robotics. Her research interests focus on the development of intelligent systems, IoT-based applications, network technologies, and automation systems that support digital transformation and smart environments. Through her academic and professional activities, she is committed to advancing applied research and fostering innovation in engineering and technology education.

Author Contributions

YSP: Research and Development, Game Design. **AM, AA, CRTP, DRT, NW:** Implementation and Initial Draft Preparation. **DE:** Supervision.

ORCID ID

YSP : https://orcid.org/0009-0008-1977-3923
AM : https://orcid.org/0009-0000-8014-4474
AA : https://orcid.org/0009-0005-2039-788X
CRTP : https://orcid.org/0009-0009-7564-1359
DRT : https://orcid.org/0009-0003-2501-9576
NW : https://orcid.org/0009-0004-2281-0896
DE : https://orcid.org/0009-0002-6502-3759

AI Usage Policy

The authors declare that artificial intelligence tools were used solely to assist with grammar correction and language editing during the preparation of the manuscript. The use of AI did not replace the authors' research, analysis, interpretation, or scientific decision-making.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data supporting the findings of this study are available from the corresponding author upon reasonable request. The Wokwi simulation project can be accessed https://s.pnj.ac.id/Rebutan.

Open Access Information

Open Access. This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. If material is not included in the article's Creative Commons license CC-BY-NC 4.0 and your intended use it, you will need to obtain permission directly from the copyright holder. You may not use the material for commercial purposes. To view a copy of this license, visit https://creativecommons.org/licenses/by-nc/4.0/

Copyright by authors@2025