MagDrop4

Leon Greiner, Murtaza Yar Hiraj, Giorgiana Rosas, Ashvin Verma
g38hob, murtazahiraj, gcrosas, ashvin.verma@berkeley.edu

1. Introduction

The MagDrop4 project brings Connect4 to life by creating an interactive, real-time playing machine that seamlessly competes against a human opponent on a physical game board.
At its core, MagDrop4 integrates one IR sensor per column to detect human moves, an AI-driven game engine to calculate optimal responses, and a magnetic piece-handling system powered by an electromagnet. A custom-built linear actuator, driven by a stepper motor and belt system, ensures precise placement of the game pieces. This project highlights the real-time integration of sensors, motor control, and AI-based decision-making, all in one compact embedded system.

2. Design and Implementation

The design of MagDrop4 is divided into four main components. First, the mechanical design focuses on building the machine's structure, designing the linear actuator, sourcing necessary parts, and creating custom 3D-printed PLA components. Second, the electrical system is developed to control the game piece handling mechanism and the linear actuator, run the game AI engine, and process all sensor inputs resulting from human interaction, all within a standalone embedded system. Third, the game AI is designed to compete against human players at a skill level where winning becomes nearly impossible. Finally, the software and machine logic are developed and implemented on the single-board computer, ensuring seamless integration of all components and smooth operation of the machine.

2.1. Mechanical Design

The entire mechanical design, including structural components, electrical parts, and 3D-printed pieces, was created and evaluated in CAD. The main structure is built using standard V-Slot aluminum profiles, which also serve as guide rails for the gantry cart of the linear actuator. The linear actuator was seamlessly integrated into the design: the gantry cart features chamfered wheels that fit perfectly into the V-slot profile, providing smooth, stable movement without wobbling. The stepper motor is mounted at one end of the rail using a motor mount and T-nuts, similar to other components such as L-connectors and custom PLA mounting parts. A toothed timing pulley is attached to the stepper motor's shaft to transfer force to a GT2 belt. At the opposite end, an idler pulley (without teeth) is mounted on a plate, which can be adjusted using T-nuts to tension the belt, ensuring precise movement of the gantry cart.

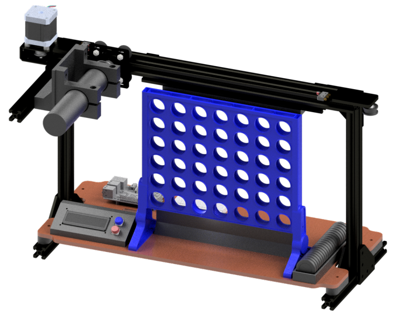
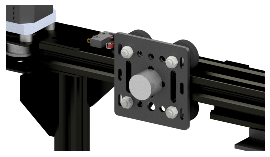


Figure 1: CAD Rendering of MagDrop4

The system's 7 IR sensors, used to detect human moves, are mounted on a 3D-printed part. This component is then securely attached to the V-slot aluminum profile using T-nuts for easy positioning and stability.
The magazine is composed of two 3D-printed PLA parts and includes a spring mechanism with a spring-loaded pusher. This setup ensures the game pieces are continuously pushed to the end of the magazine, where they can be picked up by the electromagnet. The magazine is mounted to the aluminum profile using custom 3D-printed brackets.
The game pieces are also 3D-printed to allow the insertion of magnetic A3 steel discs, enabling them to be handled by the electromagnet effectively.
Finally, the baseplate is laser-cut from plywood. It supports the Connect4 game board, the control unit with a display for game status updates, and two buttons for machine control. Additionally, it includes mounts for disc storage, the single-board computer, and the stepper motor driver, creating an organized and robust foundation for the entire system.


The gantry cart

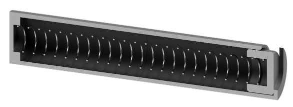

The game piece magazine

Figure 2

In total, we utilized over 50 distinct components to successfully build the Connect4 machine:

Item	**Quantity**
400-Point Breadboard	1
Aluminum Spacer	4
Belt Clamp	2
Connect 4 Game Set	1
Corner Connector 2020	1
Display	1
DRV8825 Extension Board	1
Electromagnet	1
Gantry Cart	1
GT2 Belt	1
Idler Pulley	1
Idler Pulley Plate	1
IR Sensor	7
L-Connector	7
Limit Switch	2
M2.5 Screw	8
M3 Insert x 5mm	7
M3 Screw	17
M5 Nut	12
M5 Screw	>50
M5 T-Nut	1
Metal Disc Inserts	42
Limit Switch Plate	2
Mosfet IRLZ44	1
Raspberry Pi 3 Model B+	1
Round Switch	2
Spring	1
Stepper Motor	1
Stepper Motor Driver (DRV8825)	1
Superglue	1
Timing Pulley	1
USB 2.0 A to Micro B Cable	1
USB Power Supply	1
V-Slot 20x20 100mm	4
V-Slot 20x20 200mm	1
V-Slot 20x20 300mm	3
V-Slot 20x40 500mm	1
V-Slot Cover	2
V-Slot Endcap	2
3D Printed Parts	
Base Plate Spacer	4
Control Mount	1
Disk	42
Disk Barrier	1
Disk Holder	1
Idler Pulley Overlay	1
IR Mount Plate	1
Magazine Bracket	2
Magazine Tube 1	1
Magazine Tube 2	1
Magazine Mounting Plate	1
Magazine Mounting Spacer	1
Raspberry Pi Standoff	4
Spring Pusher	1
Base Plate (Laser Cut)	1

Table 1: Purchase and Checkoff Parts List with Additional 3D Printed Parts

2.2. Electrical Design

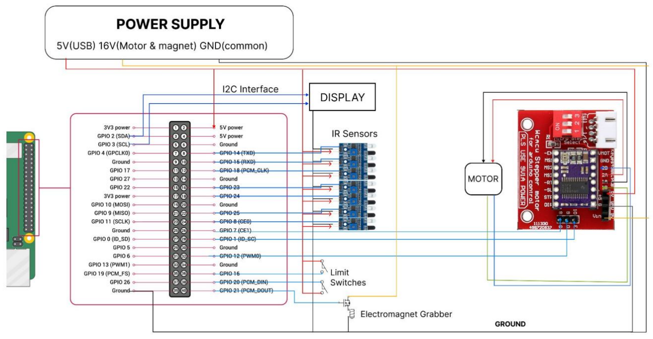


Figure 3: Electrical Diagram

The electrical system is centered around a Raspberry Pi 3 B+, which serves as the single-board computer responsible for motor control, electromagnet operation, sensor inputs, display output, the game AI, and overall machine logic. The Raspberry Pi is powered via a 5V USB power supply connected to its USB-B port. Additionally, a 12V power supply is used to drive the stepper motor and electromagnet.

The 7 IR sensors are powered by the 5V pin of the Raspberry Pi and provide outputs that are normally high but go low when triggered. The detection sensitivity can be adjusted using a small screw on each sensor. These were carefully calibrated so the sensors trigger when a white, highly reflective game piece is placed into a column but remain unresponsive to hands or other obstructions near the game board. This fine-tuning was essential for reliable operation.

The two limit switches are connected to ground and to GPIO pins with internal pull-up resistors enabled. When triggered, the switches pull the GPIO pins low; otherwise, they remain high. This setup is identical for the two buttons used for human inputs.

The stepper motor is connected to the DRV8825 motor driver via its four coil wires. The motor driver receives control signals from the Raspberry Pi and converts them into current signals to drive the motor smoothly and accurately. To avoid skipped steps or overheating, the motor current is carefully adjusted using a potentiometer on the driver, following the specifications in the datasheet. The motor driver itself is mounted on an extension board for simplified wiring. The motor is controlled through three GPIO pins:

1. Enable: Turns the motor on (low) or off (high).
2. Direction: Sets the motor direction (high for left, low for right movement of the gantry cart).
3. Step: Moves the motor one step each time the pin transitions from low to high.

Microstepping is configured to 1/8 step resolution to ensure smooth and precise movement of the gantry cart.

The electromagnet operates on 12V and is controlled by a single GPIO pin. To regulate the 12V power, an N-channel logic-level MOSFET (IRLZ44) is used, along with a 10kΩ pull-down resistor and a flyback diode. The diode prevents damage to the motor driver and MOSFET caused by back EMF when the electromagnet is turned off.
The display operates at 3.3V and is connected to the Raspberry Pi via I2C using two GPIO pins. This connection allows the display to show real-time game status and other prompts during gameplay.

2.3. Game AI

At the heart of the AI is the Minimax algorithm, which explores all possible moves up to a specified search depth (DFS) and evaluates each board state based on a scoring function. For instance, at a search depth of 2, the algorithm first evaluates all 7 possible moves (assuming no columns are full), and then considers all subsequent moves by the opponent, analyzing a total of $7^2 = 49$ possible paths. Then the move that guarantees the best outcome, based on the scoring function, is selected.
A key assumption of the Minimax algorithm is that both the machine and the human opponent play optimally. This means the AI not only chooses the best possible move for itself, but also anticipates that the human will respond with their best move. For example, if a particular AI move would allow the human to win in one move, the AI will avoid that option, assuming the human will capitalize on the opportunity.
In reality, however, humans are prone to errors, which are unpredictable and cannot be accounted for by the algorithm. Despite this limitation, the Minimax algorithm performs exceptionally well in practice against human players, not just against other AI's.

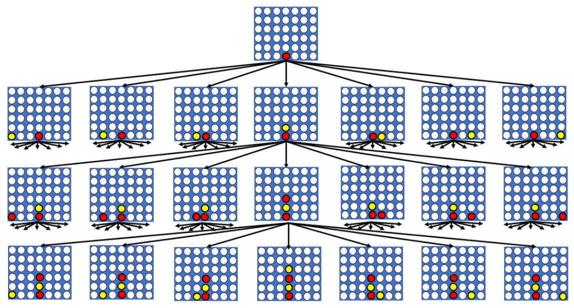


Figure 4: Partial game tree, with Minimax searching up to depth 3 [1]

However, this approach becomes computationally expensive as the search depth increases. Alpha-Beta pruning improves the efficiency of Minimax by eliminating branches that cannot influence the outcome, significantly reducing the number of node evaluations. The diagram in Figure [5] compares the number of nodes evaluated at increasing search depths for Brute Force Minimax and average nodes evaluated by our optimized Alpha-Beta Pruning algorithm in Connect4, since the efficiency is highly dependent on the current game state and the heuristic. In the worst case, Alpha-Beta pruning evaluates as many nodes as Brute Force, but in reality, the efficiency gains increase exponentially with the depth.

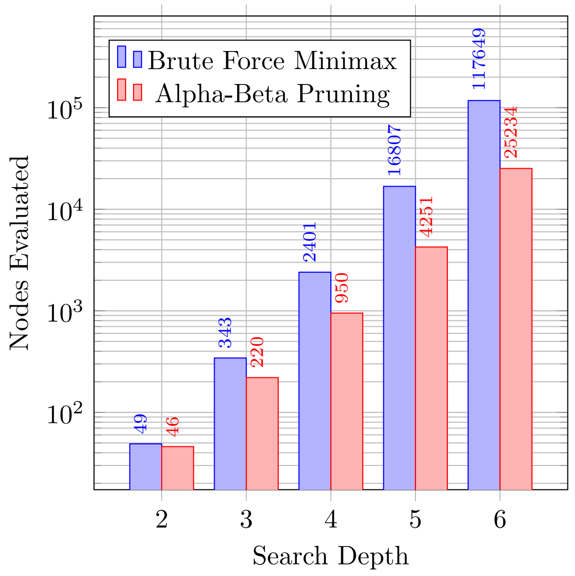


Figure 5: Comparison of Nodes Evaluated: Brute Force Minimax vs Alpha-Beta Pruning

The heuristic/scoring function evaluates windows of four consecutive cells (horizontal, vertical, and diagonal) and scores them based on the number of AI and human pieces present. Additionally, the AI incentivizes central play, as pieces placed in the center column create more opportunities to connect in multiple directions.

- 4 AI pieces (winning move): `+100`
- 3 AI pieces: `+10`
- 2 AI pieces: `+5`
- For every piece at the middle column: `+6`
- 4 human pieces (immediate loss): `-100`
- 3 human pieces: `-8`

The implementation was done in Python and relies on the following libraries:

- `Numpy`: Used for efficient creation, manipulation, and copying of the game board, which is represented as a 2D matrix.
- `random.shuffle`: Ensures variety in the AI's behavior by randomizing valid moves of equal scores. This avoids predictability and enhances the challenge for the human player.

2.4. Machine Logic

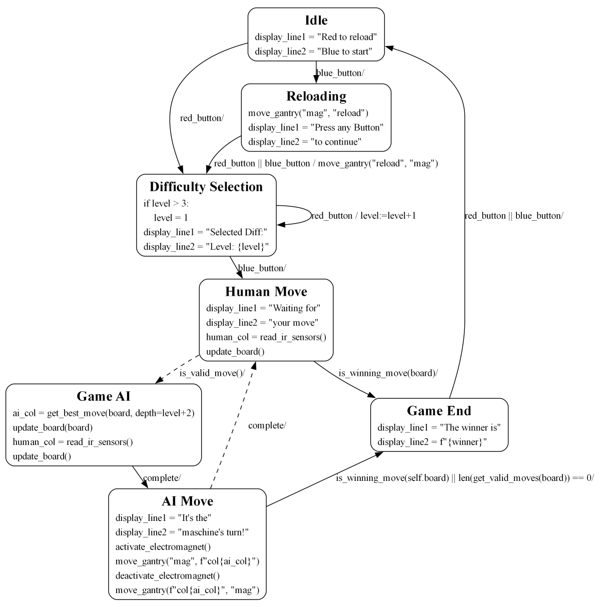

Idle
display_line1 = "Red to reload"
display_line2 = "Blue to start"
blue_button/
Reloading
move_gantry("mag", "reload")
display_line1 = "Press any Button"
display_line2 = "to continue"
red_button/
red_button || blue_button / move_gantry("reload", "mag")
Difficulty Selection
if level > 3:
level = 1
display_line1 = "Selected Diff:"
display_line2 = "Level: {level}"
red_button / level:=level+1
red_button || blue_button/
blue_button/
Human Move
display_line1 = "Waiting for"
display_line2 = "your move"
human_col = read_ir_sensors()
update_board()
is_valid_move()/
is_winning_move(board)/
Game AI
ai_col = get_best_move(board, depth=level+2)
update_board(board)
human_col = read_ir_sensors()
update_board()
complete/
Game End
display_line1 = "The winner is"
display_line2 = f"{winner}"
complete/
is_winning_move(self.board) || len(get_valid_moves(board)) == 0/
AI Move
display_line1 = "It's the"
display_line2 = "maschine's turn!"
activate_electromagnet()
move_gantry("mag", f"col{ai_col}")
deactivate_electromagnet()
move_gantry(f"col{ai_col}", "mag")

Figure 6: Simplified illustration of the system's FSM

The machine logic, implemented as a Finite State Machine (FSM) in Python, is shown in a simplified version in Figure [6]. The FSM seamlessly integrates the game AI and all other sub-functions saved under `modules`. The complete FSM implementation can be found in the GitHub repository under the main Python script `code/magdrop.py`.
The system starts in the *Idle* state, where all variables are initialized and set to reasonable default values. Additionally, a *Reloading* state moves the gantry cart to the rightmost position, allowing space to reload the magazine. The machine also keeps track of the number of discs required after a previous game, displaying this information on the LCD screen. For clarity, these details are not included in Figure [6]. A *Difficulty Selection* state enables the user to cycle between three AI difficulty levels: Medium, Hard, and Unbeatable, which correspond to a search depth of 3, 4, and 5 of the Minimax algorithm. When playing a game, the machine cycles through three main in-game states, until a terminal game situation is reached and the game ends.

1. *Human Move:* The system polls through the IR sensor pins to detect the player's move. Since the machine performs only one task at a time, polling is sufficient, and advanced techniques such as interrupts are not necessary.

2. *Game AI:* The AI computes the optimal move using the `get_best_move` function from `modules.game_ai`.

3. *AI Move:* The machine executes its move by controlling the electromagnet to grab the game piece and the stepper motor to place it in the correct column. The stepper motor's motion is achieved by toggling the step pin high and low with a delay of 0.0001s. With the used microstepping of 1/8, this results in a step every 0.8ms and a rotational speed of 187.5 RPM (based on 400 steps per motor rotation). To ensure precise game piece placement, a function was implemented to store the step distances from the leftmost position (at the magazine, where the gantry hits the left limit switch) to all seven columns. These distances were carefully determined through testing. Sleep delay was used to ensure a smooth process of grabbing, moving, and releasing the game pieces.

The following Python libraries were utilized in the project, in addition to the ones used for the game AI:

- `RPi.GPIO`: For handling the GPIO pins on the Raspberry Pi.
- `time`: Used for delays to control the stepper motor, debounce buttons, and manage the overall machine processes.
- `qwiic_serlcd`: Simplifies integration with the SparkFun SerLCD via the I2C interface, enabling display output for game states and user prompts.

This modular functions structure, with one main FSM process logic, ensures that the IR sensor, buttons, limit switch, and display handling, as well as the electromagnet and motor control, are well-structured, efficient, and easy to maintain.

3. Evaluation

3.1. Challenges

Throughout the project, we encountered various challenges, some anticipated and others unexpectedly complex. These challenges spanned the areas of software development, mechanical design, and system integration.
One of the first challenges was the implementation and development of the game AI. As we waited for the ordered mechanical parts to arrive, we utilized this time to develop a fully functional Connect 4 game engine, which we tested virtually. While the Minimax algorithm with Alpha-Beta Pruning is a widely used technique for game AI, and resources on its implementation are readily available, we still needed to code it from scratch. The AI had to meet specific requirements, such as seamless integration with the machine's overall process flow and the ability to run efficiently on an embedded system with limited computational resources.
The mechanical design, including mounting the IR sensors and implementing the belt-driven linear actuator, was relatively straightforward. However, designing the magazine proved to be more challenging. Since the magazine was constructed from two PLA-printed tubes glued

together, precision was critical to ensure that the spring-loaded pusher moves the game discs reliably without jamming, while also avoiding excessive looseness, which could cause the discs to twist and get stuck. Additionally, the spring required careful tuning, it had to be strong enough to push the last disc to the grabbing point but not so strong that the electromagnet couldn't pull the first game piece. Lastly, the magazine had to be designed for easy reloading by hand, ensuring usability for human operators.

Integrating all components into a cohesive, enclosed embedded system presented significant challenges. Unexpectedly, controlling the stepper motor became a tricky task. Early on, we fried two motor drivers, likely due to electromagnetic back EMF, until we mitigated the issue by using a protective diode. Adjusting the motor's current flow via the driver's potentiometer was another critical step, which we hadn't initially anticipated. Proper calibration of the current was necessary to avoid overheating the motor while still delivering sufficient torque to prevent skipped steps. Additionally, we encountered an issue where the game pieces would remain stuck to the electromagnet even after it was turned off, likely due to residual magnetism. We addressed this by gluing a thin piece of paper onto the electromagnet, which prevented direct contact between the magnet and the steel discs. However, this solution also slightly reduced the electromagnetic force, impacting the reliability of pulling game pieces out of the magazine.

In contrast, IR sensor calibration turned out to be straightforward, as did most of the wiring, despite the complexity introduced by over 50 individual wires. Careful planning and organization were key to ensuring that all components, from sensors to actuators, operated seamlessly within the system.

3.2. Results

The final results of the project were highly promising, and the machine was successfully able to play Connect4 against human players in a convenient and engaging way. Demonstration videos showcasing the machine in action can be found here. The system provided a lot of entertainment to those who attempted to win against it. Notably, no one managed to defeat the machine in *Hard* or *Unbeatable* mode, and only two victories were recorded in *Medium* mode out of countless attempts. This highlights a need for improved game balancing in future work. However, we determined that reducing the search depth below 3 was not a viable option. During testing, depths lower than 3 made the machine noticeably "dumb," as it failed to anticipate traps or set up strategic plays. A better approach for future work could involve maintaining a depth of 3 while introducing a degree of randomness to make gameplay more balanced and the AI beatable.

The IR sensors proved to be highly robust in detecting the correct columns. They effectively ignored hands or other objects in front of the board and only triggered when an actual game piece was placed inside a column. This feature significantly improved usability. However, one minor issue was observed: if a game piece was held at the top of a column without fully releasing it, the corresponding IR sensor could still be triggered. This could be addressed in future iterations by designing a custom Connect4 game board with a larger frame, preventing premature sensor activation.

The machine's overall performance during gameplay was both accurate and efficient. In *Hard* mode, the time between a human move being detected and the machine completing its move (including AI computation and piece placement) was, in most cases, under 5 seconds. This ensured a smooth and responsive gaming experience.

On the mechanical side, we faced limitations with the magazine design. The electromagnetic force was slightly too weak, and the spring was too strong, allowing only 7 game discs to be loaded at a time for reliable operation. In future work, this issue could easily be resolved by using a weaker spring and slightly thicker metal discs. The spring preload was already sufficient to push the last game piece to the grabbing point, so a weaker spring would allow for all 21 discs to be loaded into the magazine, greatly improving convenience and functionality.

Overall, the machine performed reliably and efficiently, with clear opportunities for further refinement in future iterations to enhance both gameplay and mechanical robustness.

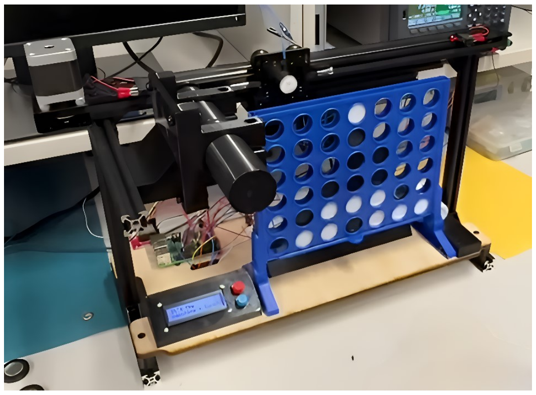


Figure 7: A snapshot of a game in progress

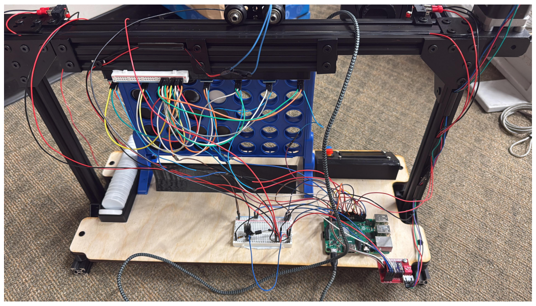


Figure 8: The wiring at the back of MagDrop4

Key Course Concepts

The following course topics were utilized, implemented, and applied in this project, as already previously discussed:

1. **Ch.3: Discrete Dynamics**
   Implemented as a Finite State Machine (FSM) to manage system states such as Idle, Human Move, AI Move, and Game End.
2. **Ch.4: Timed Automata**
   Timers were used in the system's FSM to ensure smooth process of the machine.
3. **Ch.5: Composition of State Machines**
   Multiple state machines were composed to handle user input, AI decision-making, and actuator control seamlessly.
4. **Ch.7 Sensors and Actuators**
   Usage of techniques like polling and debouncing of IR sensors, limit switches, and buttons, alongside control of actuators like the stepper motor and electromagnet.
5. **Ch.10: Input, Output and Timers**
   GPIO pins were used for sensor input and actuator control, with timers ensuring smooth motor operations and systems process.

Project Resources and Demonstration

All project files, including documents, images, source code, CAD files, and other relevant materials, are available in the GitHub repository:
https://github.com/Loneli999/MagDrop4.git

Demonstration videos showcasing the project in action can be viewed on YouTube:
https://youtu.be/q6UGUV51KW0
https://youtube.com/shorts/QP-eAZoq4xQ

References

[1] T. Ngo, *Can you create a strategic connect 4 ai opponent?* Accessed: 2024-12-15, 2024. [Online]. Available: https://www.sciencebuddies.org/science-fair-projects/project-ideas/ArtificialIntelligence_p014/artificial-intelligence/alpha-beta-connect4.