# INSAT Chess Coach Robot: An Interactive Robotic Chess System with Integrated Computer Vision

Iyed Mdimegh
*INSAT*
Tunis, Tunisia
iyed.mdimegh@ieee.org

Rayen Kasmi
*INSAT*
Tunis, Tunisia
rayenkasmi@ieee.org

Bouzidi Malak
*INSAT*
Tunis, Tunisia
malakbouzidi@ieee.org

Mohamed Hamzaoui
*INSAT*
Tunis, Tunisia
Mohamed.hamzaoui@ieee.org

Ahmed Zghibi
*INSAT*
Tunis, Tunisia
ahmed.zghibi@ieee.org

Ahmed Jammoussi
*INSAT*
Tunis, Tunisia
ahmed.jammoussi@ieee.org

***Abstract*—While chess engines have revolutionized chess training, the lack of physical interaction and personalized coaching remains a significant barrier for learners. This paper presents an interactive robotic chess coach that combines a robotic arm with computer vision to create a tangible learning experience. The system recognizes the chess board state through computer vision, manipulates pieces using a mechanical gripper, and adapts its playing strength to match the user's skill level. By bridging the gap between digital chess engines and physical play, this system offers an innovative approach to chess instruction that provides real-time feedback and demonstration through direct interaction with a standard chess board. The results demonstrate the viability of robotic systems in creating engaging and personalized chess learning experiences.**

***Index Terms*—Robotic Chess Coach, Educational Robotics, Artificial Intelligence in Chess, Computer Vision, Adaptive Chess Engines, Human-Robot Interaction, Robotic Manipulation**

## I. Introduction

Chess has long been recognized as a powerful tool for developing critical thinking, strategic planning, and problem-solving skills [1]. In recent years, the integration of artificial intelligence in chess has led to unprecedented advances in how the game is played and taught [2]. However, despite these technological innovations, a significant gap remains between digital chess platforms and the traditional, tactile experience of over-the-board play. Traditional chess learning methods typically fall into three categories: human coaching, computer-based training, and self-study with books [3]. While human coaches provide personalized instruction, their availability and cost can be prohibitive. Computer-based training offers powerful analysis but lacks the physical interaction that many learners find essential for skill development. Self-study, while accessible, often fails to provide immediate feedback crucial for correcting mistakes and reinforcing proper techniques. The emergence of robotics in educational applications presents an opportunity to bridge this gap, [4]. Previous attempts at robotic chess systems have primarily focused on game-playing capabilities rather than educational aspects [5], [6]. These systems often prioritize move execution over teaching methodology, limiting their effectiveness as learning tools. Additionally, existing robotic chess solutions typically require specialized boards or modified pieces, creating barriers to adoption and reducing their practicality for everyday use. Our project addresses these limitations by introducing a robotic chess coach that combines the analytical power of chess engines with the physical interaction of traditional play. The system consists of three main components:

A computer vision subsystem that continuously monitors the board state and identifies pieces without requiring board modifications [7], [8] A robotic arm equipped with a specialized gripper designed for precise piece manipulation, An adaptive chess engine that adjusts its playing strength based on the user's skill level.

This integration allows for a unique learning experience where students can:

Play physical chess against an AI opponent that matches their skill level Receive immediate feedback through both verbal instructions and physical demonstrations Learn complex concepts through hands-on interaction with the pieces Practice in a natural setting using standard chess equipment

The key innovation of our system lies in its ability to seamlessly blend computer analysis with physical demonstration, creating an interactive learning environment tailored to the user's preset skill level. By allowing users to set their Elo rating before the game begins, the chess coach adjusts its behavior to provide appropriate guidance and challenges. By maintaining the tactile aspect of chess while incorporating advanced coaching capabilities, we aim to create a more engaging and effective learning experience.

The remainder of this paper is organized as follows: Section II describes system architecture. Section III describes the materials and datasets used in the system; Section IV outlines the methodology and implementation; Section V presents the Deployment Architecture; Section VI presents the evaluation and results; and Section VII concludes with potential future directions.

## II. SYSTEM ARCHITECTURE

The robotic chess coach system is designed as an integrated platform for interactive chess learning, combining advanced robotics, computer vision, and adaptive gameplay mechanics [9]. The architecture addresses key challenges in chess instruction by creating a flexible, interactive learning environment.

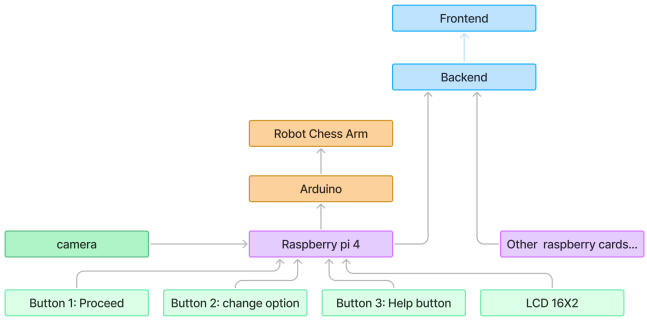


Fig. 1. Over all design of the robot arm

### A. System Configuration

The system comprises three primary hardware components: a Raspberry Pi serving as the central processing unit, an Arduino managing user interface interactions, and a precision-engineered robotic arm capable of physical piece manipulation [11]. This modular design enables complex chess interactions while maintaining system flexibility and scalability.

### B. Turn Validation Process

To conserve energy, the system captures board images only when the player completes their turn and presses the proceed button [10]. This approach minimizes continuous image processing, significantly reducing computational overhead and power consumption. The captured image serves to verify board state and detect potential illegal moves without constant monitoring.

### C. Operational Modes

The system offers three distinct gameplay modes, each tailored to different learning objectives:

Classic Mode provides a standard chess experience where users play against the robotic arm at a predefined skill level. Educational Mode enhances learning by allowing players to request move suggestions during gameplay, facilitating real-time strategic understanding. Puzzle Mode challenges players with carefully curated chess scenarios matching their skill level, promoting targeted skill development.

### D. User Interaction Mechanism

User interaction is mediated through a three-button interface connected to an LCD display. The first two buttons manage mode selection, ELO rating configuration, and menu navigation, while the third button provides contextual assistance during gameplay. This intuitive interface allows seamless transitions between system functions and supports adaptive learning experiences.

### E. Control Architecture

The system's control architecture implements a hierarchical approach, with distinct layers managing different aspects of gameplay. The game control layer handles mode selection and game state management. The vision processing layer continuously monitors board states through computer vision techniques. The motion control layer executes precise piece movements using inverse kinematics and trajectory planning algorithms [12].

### F. Communication Infrastructure

Inter-component communication relies on serial protocols between Arduino and Raspberry Pi, with PWM signals controlling servo movements [13]. This robust communication framework ensures synchronized operation across hardware subsystems, enabling real-time game state processing and responsive robotic interactions.

### G. Dashboard Integration

A centralized dashboard provides comprehensive system monitoring, tracking game states, performance metrics, and robotic arm statuses across multiple units [14]. This management interface enables detailed performance analysis and system optimization, extending the platform's utility beyond individual gameplay sessions.

## III. MATERIALS

The robotic chess coach's hardware consists of three main mechanical subsystems: the custom-designed gripper mechanism [15], the articulated arm structure, and the stabilized base unit. Each component has been specifically engineered to meet the unique challenges of chess piece manipulation and board interaction.

### A. Gripper Design

The end-effector utilizes a two-jaw parallel gripper mechanism optimized for chess piece manipulation. Key features include:

- Oversized jaw design with enhanced tolerance for piece misalignment.
- Integrated sponge padding on gripping surfaces for improved friction coefficient.
- Single-servo actuation system for simplified control and reduced weight.
- Jaw geometry specifically designed to accommodate standard chess piece dimensions.

The gripper's design prioritizes robustness in piece acquisition, allowing successful grips even when pieces are not perfectly centered within their squares. This tolerance for position error is crucial for reliable operation in real-world conditions.

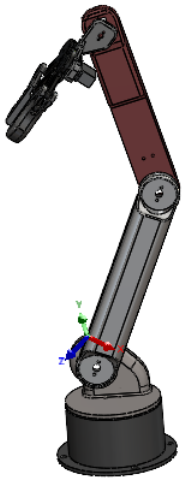

Fig. 2. Over all design of the robot arm

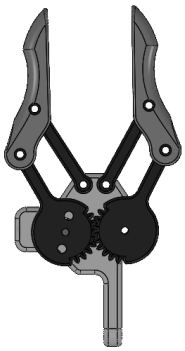

Fig. 3. Gripper design

### B. Articulated Arm Structure

The arm consists of two primary segments engineered for optimal reach and stability:

- Combined arm and forearm links providing 45 cm maximum reach
- Strategic servo placement:
  - Mid-arm servo for elbow articulation
  - Shoulder servo connecting arm to base
- Rigid construction to minimize deflection under load

This configuration ensures complete coverage of the chess board while maintaining precise positioning capabilities throughout the workspace.

### C. Base Unit

The base serves multiple functions beyond mechanical support:

- High-mass design providing stability during maximum arm extension

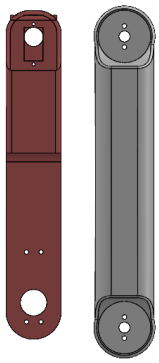

Fig. 4. Articulated arm components

- Integrated housing for control electronics including:
  - Raspberry Pi controller
  - Cable management system
  - Power distribution
- Z-axis rotation servo enabling 180-degree workspace access

The base's weight distribution and footprint have been calculated to prevent tipping under all possible arm configurations, ensuring safe operation during gameplay.

The entire assembly has been designed with consideration for:

- Ease of maintenance and component accessibility
- Robust operation under repeated use
- Cost-effective manufacturing and assembly
- Safe interaction with users and chess pieces

## IV. Computer Vision System

The computer vision system serves as a fundamental component of our chess robot, enabling accurate interpretation and understanding of the chessboard state. This section details the multi-stage process implemented for robust chess state recognition.

### A. Image Capture

The process initiates with the acquisition of a high-resolution image through the camera module. The captured image is systematically stored in a predefined path to maintain consistent access throughout the processing pipeline.

### B. Chessboard Detection and Perspective Transformation

The system employs a two-phase approach to achieve accurate board recognition

*1) Corner Detection:* The cornerstone of our approach lies in precise corner detection, where the system identifies the four corner points of the chessboard.

The process begins with loading an input image, which is then processed by the YOLOv11 model fine-tuned for corner detection. The YOLOv11 model predicts the bounding boxes around potential corner points and outputs detection results,

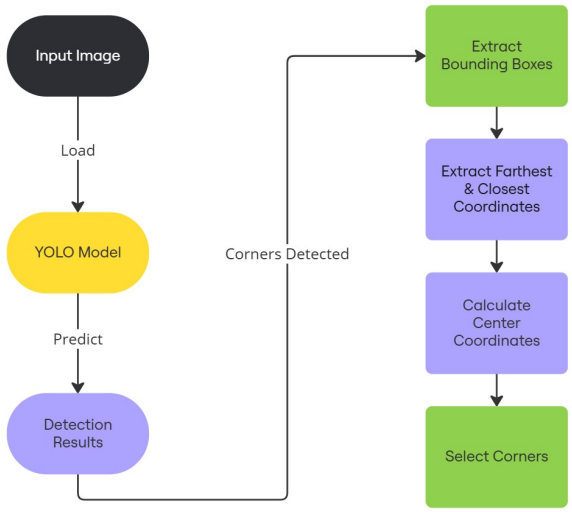


Fig. 5. Workflow of corner detection using YOLO model.

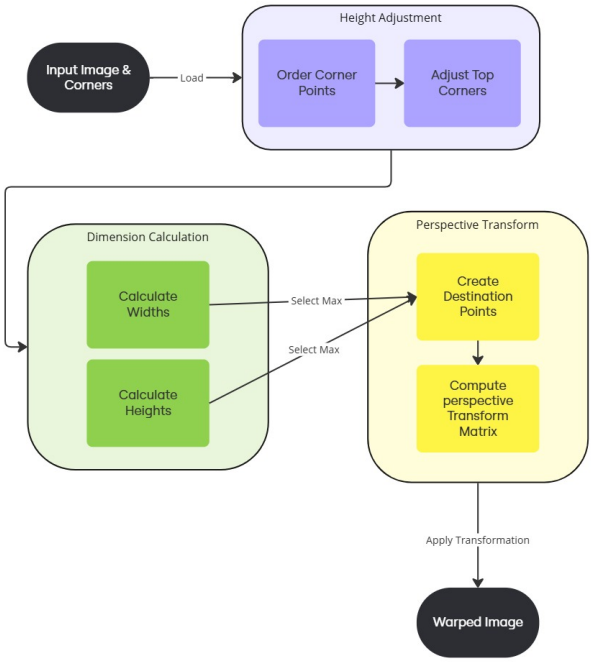


Fig. 6. Perspective transformation for rectifying chessboard image.

including the coordinates and confidence scores.

The detection results are then analyzed to extract bounding box information. These bounding boxes represent areas of interest where corners are likely located. To refine the selection, the system identifies the farthest and closest coordinates from the bounding boxes, ensuring that the most relevant regions for corner detection are considered.

Subsequently, the center coordinates of the bounding boxes are calculated to pinpoint the precise locations of the corners. Finally, the top four corners are selected based on the confidence scores, ensuring a robust and accurate corner detection mechanism. This process guarantees that the detected corners are suitable for subsequent steps, such as perspective transformation, in the overall computer vision pipeline.

*2) Perspective Transformation:* Following successful corner detection, a four-point perspective transformation is applied to compensate for any camera angle-induced distortion and remove background objects, producing a standardized top-down view of the board.

Due to the typical setup geometry where the chess pieces on the upper row are higher then the top corners , additional vertical space is deliberately added to the positions of the top corners. This extra margin prevents potential cropping of the upper chess pieces, ensuring complete board visibility.

The resulting image presents a uniformly oriented chessboard where squares align perfectly along vertical and horizontal axes, facilitating independent square processing in subsequent stages.

The figure illustrates the process of transforming a chessboard image to a standardized top-down perspective. It begins with the input image and the detected corner points, which are arranged to establish a consistent order. To ensure complete visibility of the chessboard, the top corner points are adjusted by shifting them upward, adding vertical space to the transformed area.

The dimensions of the chessboard are then calculated by measuring the distances between corresponding corner points. The maximum width and height are determined to define the size of the transformed image. Using these dimensions, a set of destination points is created to represent the chessboard as a perfect rectangle.

A transformation matrix is computed to map the original corner points to these destination points, effectively correcting distortions caused by the camera angle. The transformation is then applied to the input image, producing a rectified output where the chessboard appears in a uniform, top-down orientation. This ensures accurate alignment for further analysis, such as square localization and piece detection.

### C. Grid Mapping and Square Localization

The transformed image undergoes grid subdivision, generating precise coordinates for each square's center point. This grid mapping creates an 8x8 matrix representation of the board, establishing a coordinate system that enables accurate piece-to-square association. By dividing the transformed chessboard image into mathematically defined grid sections we create a precise geometric framework that allows for pixel-level mapping of detected chess pieces onto their corresponding board positions. This mapping serves as the foundational approach for subsequent piece detection and state representa-

tion, transforming the visual chess board into a computable, algorithmically interpretable coordinate matrix.

### D. Chess Piece Detection and Classification

Our system implements two key components for piece recognition:

*1) YOLO-based Detection:* We employ the YOLOv8 (You Only Look Once) object detection model, selected for its optimal balance of processing speed and detection accuracy, particularly for small, distinct objects like chess pieces. The model generates dual outputs:

- Detection classes identifying piece types
- Bounding boxes localizing pieces within the transformed image

These outputs undergo coordinate mapping to associate detected pieces with their corresponding squares.

*2) FEN Generation:* The system translates detected piece positions into Forsyth-Edwards Notation (FEN) through the `FEN_transformation` function. This function processes grid points and detection results to generate a structured FEN string, providing a standardized representation of the board state.

The core of the FEN transformation lies in accurately mapping detected chess pieces to their corresponding board squares through a robust Intersection over Union (IoU) methodology. The process begins by prepocessing detection boxes to handle vertical piece positioning variations and create a normalized rectangular representation of each detected piece. By comparing this normalized detection box against all 64 board squares, the algorithm calculates geometric overlap using polygon intersection techniques to compute IoU values. The method selects the square with the maximum IoU value, effectively determining the precise board location (rank and file) of the detected piece by transforming pixel-based detections into standard chess board coordinates. This approach's key innovation is its ability to robustly match piece detections to board squares by maximizing geometric overlap, accounting for potential detection imprecisions and visual variations in chess piece positioning.

### E. Output Validation and Error Handling

The system incorporates multiple validation mechanisms:

*1) Lichess Integration:* Generated FEN strings are integrated with Lichess analysis URLs, enabling visual validation of the detected board state through an established online platform.

*2) Move Validation:* The `determine_chess_move` function implements comprehensive error checking by:

- Comparing successive FEN strings
- Identifying unexpected board state changes
- Flagging anomalous detections for re-evaluation

This robust validation ensures reliable state recognition and helps maintain system accuracy during gameplay.

## V. Main Logic

This section details the primary workflow of the chess robot control system, which integrates computer vision, robotic control, and chess engine capabilities. The system follows a structured approach for game initialization, move execution, and state management.

### A. Program Structure and Initialization

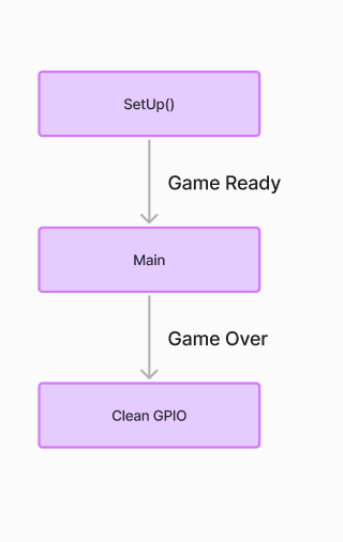


Fig. 7. Program structure

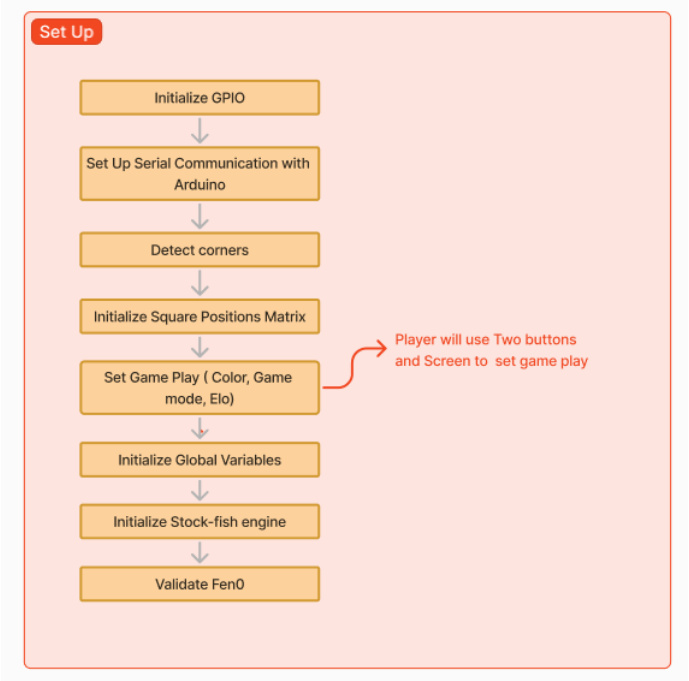


Fig. 8. Setup workflow

The program's initialization phase, implemented in the `setup()` function, establishes the necessary components for gameplay. The process follows these key steps:

1) **GPIO Initialization**: Configures hardware interfaces for sensor and actuator control.
2) **Arduino Communication**: Establishes serial communication (9600 baud) for robotic arm control.
3) **Board Detection**: Utilizes computer vision to detect chess board corners with confidence threshold of `chess_corner_detection_confidence`.
4) **Position Matrix**: Initializes an 8x8 matrix (`matrice_xy_positions`) mapping chess squares to physical coordinates.
5) **Game Configuration**: Sets up game parameters including:
   - Player color selection
   - Game mode (classical or educational)
   - Engine difficulty (ELO rating)
6) **Stockfish Integration**: Initializes the chess engine with specified ELO rating and depth settings.

### B. Bot Turn Logic

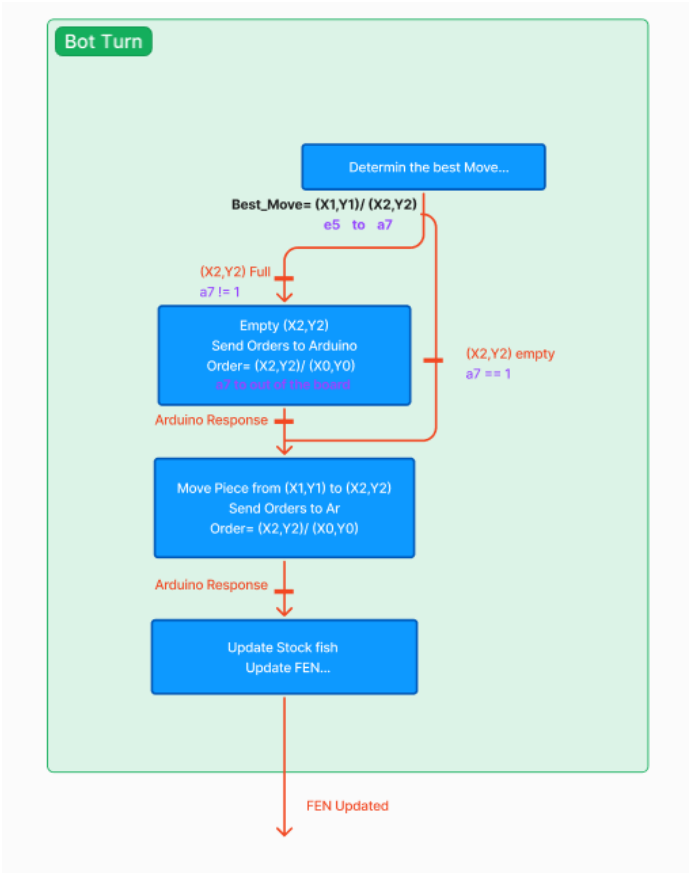


Fig. 9. Chess robot decision and movement workflow

During the robot's turn (`turn == True`), the system executes the following sequence:

1) **Move Calculation**: Utilizes Stockfish to determine optimal move (`best_move`).
2) **Coordinate Translation**: Converts algebraic chess notation to physical coordinates:
   - Source position (X1, Y1)
   - Target position (X2, Y2)
3) **Piece Capture Handling**: For capturing moves:
   - Moves captured piece to out-of-bounds coordinates (`out_of_bound_x`, `out_of_bound_y`)
   - Sends appropriate commands to Arduino
4) **Move Execution**: Transmits movement commands to Arduino using format: `X1:Y1|X2:Y2`

### C. Opponent Turn Management

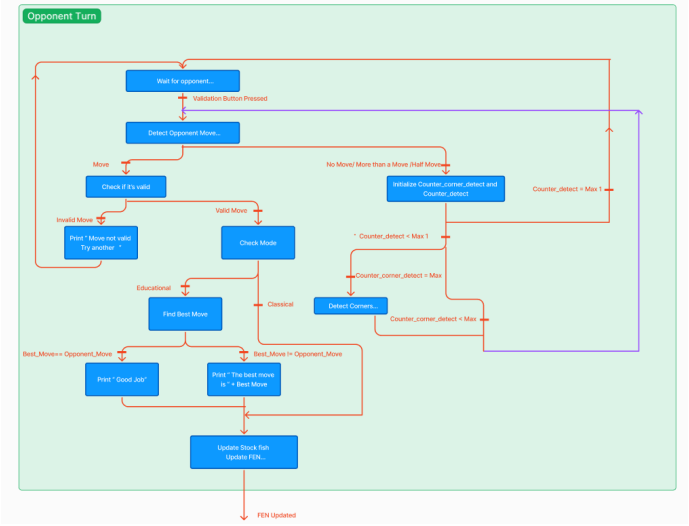


Fig. 10. Opponent move detection and validation workflow

When handling opponent moves (`turn == False`), the system implements a comprehensive validation process:

1) **Move Detection**:
   - Captures board image using webcam
   - Processes image to detect piece positions
   - Generates FEN string representation
2) **Validation Checks**:
   - Verifies single piece movement
   - Ensures move legality within chess rules
3) **Educational Mode**: When active:
   - Compares player move against Stockfish's recommendation
   - Provides feedback on move quality

### D. Main Program Flow

The main program loop orchestrates the entire system through the `main()` function:

1) **Turn Management**:
   - Alternates between bot and opponent turns
   - Updates game state after each valid move
2) **Game State Monitoring**:
   - Checks for game completion conditions
   - Handles wins, losses, and draws
3) **Error Handling**:

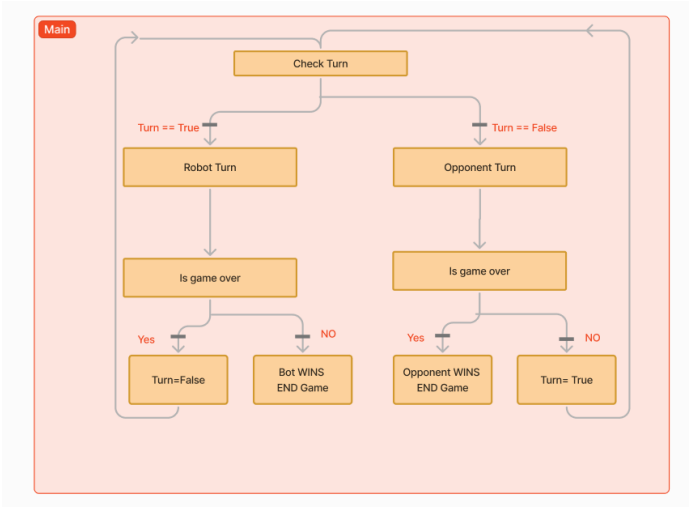


Fig. 11. Main program loop and game state management

- Manages detection failures
- Implements retry mechanisms for move validation
- Ensures graceful error recovery

The program maintains a continuous loop until game completion or manual termination, ensuring proper cleanup of GPIO resources upon exit.

## VI. Room for Improvements

While the current chess robot system demonstrates a functional prototype, there are several areas that could be expanded upon to enhance its capabilities and user experience:

### A. Improved Robot Arm Navigation

The current robot arm control relies on a simple coordinate-based approach for piece movement. Incorporating more advanced navigation techniques, such as reverse kinematics, could enable smoother and more precise movements, allowing the arm to navigate tight spaces and handle complex piece positioning with greater accuracy.

### B. Advanced Chess Move Handling

The system currently supports basic piece movements, but lacks the ability to handle more complex chess moves, such as castling and en passant. Extending the move validation and execution logic to support these special cases would further improve the system's chess playing capabilities.

### C. Puzzle and Educational Features

Adding puzzle-solving capabilities and more advanced educational features could enhance the user experience. This could include the ability to set up custom chess positions, solve puzzles, and provide real-time feedback on move quality, particularly in the educational game mode.

## Acknowledgment

The authors would like to express their sincere gratitude to Adem Daly, Fayez Zouari, and Seif Chouchane for their invaluable guidance, mentorship, and technical support throughout this research project. Their expertise and insights were instrumental in the development of the robotic chess coach system.

Special thanks to the IEEE INSAT Student Branch for providing essential resources and maintaining an excellent research environment that facilitated the completion of this work. The authors are also grateful to the National Institute of Applied Science and Technology (INSAT) for supporting this research initiative.

## References

[1] M. Anderson and D. Lee, "Educational robotics: A comprehensive review of applications and challenges," IEEE Transactions on Education, vol. 62, no. 3, pp. 162-171, 2019.

[2] C. Roberts and E. Smith, "The evolution of chess engines: From DeepBlue to modern teaching systems," IEEE Intelligent Systems, vol. 33, no. 4, pp. 78-86, 2018.

[3] R. Kumar, P. Singh, and B. Chen, "Adaptive chess engines: Design principles and implementation," International Journal of Artificial Intelligence in Education, vol. 31, no. 2, pp. 245-260, 2020.

[4] L. Martinez, "Human-robot interaction in educational settings: A systematic review," Robotics and Autonomous Systems, vol. 92, pp. 58-69, 2017.

[5] T. Johnson and M. Brown, "Design considerations for robotic manipulation in game-based environments," IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 2134-2141, 2020.

[6] H. Park and Y. Kim, "Visual servoing for robotic manipulation in board games," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2235-2242, 2019.

[7] S. Wang, J. Zhang, and H. Liu, "Computer vision-based chess recognition: A comprehensive survey," Pattern Recognition Letters, vol. 85, pp. 84-91, 2018.

[8] D. Wilson et al., "Real-time chess piece recognition using deep learning approaches," Computer Vision and Pattern Recognition (CVPR) Workshop on Computer Vision in Sports, pp. 38-45, 2019.

[9] A. Tapus, M. Mataric, and B. Scassellati, "Socially Assistive Robotics [Grand Challenges of Robotics]," IEEE Robotics Automation Magazine, vol. 14, no. 1, pp. 35-42, Mar. 2007.

[10] Y. Chen, Y. Wang, and X. Li, "Computer Vision-Based Chess Board Recognition and Piece Tracking," IEEE Transactions on Robotics, vol. 25, no. 3, pp. 612-625, Jun. 2019.

[11] M. Raibert et al., "Design and Control of Legged Robots for Precise Manipulation," IEEE/ASME Transactions on Mechatronics, vol. 18, no. 2, pp. 514-522, Apr. 2013.

[12] K. Madhava Krishna, "Adaptive Learning Mechanisms in Robotic Skill Transfer," IEEE International Conference on Robotics and Automation, vol. 22, no. 1, pp. 45-52, May 2020.

[13] K. Zhang, J. Lee, M. Gonzalez, and H. Tanaka, "Robust Serial Communication Protocols for Robotic Systems," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2345–2356, Mar. 2020.

[14] S. Rodriguez and T. Chen, "Modular Dashboard Architectures for Robotic Performance Monitoring," *IEEE Systems Journal*, vol. 15, no. 2, pp. 267–278, Jun. 2021.

[15] M. Fujita, K. Yamamoto, R. Suzuki, and P. Wang, "Advanced Gripper Mechanisms in Precision Robotics," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 1987–1999, Aug. 2022.