Robotics Programming Project

Presented by:
Akhilesh Negi, 221AI008
Gagan Deepankar V, 22AI019

Problem Statement

To develop a user-centric robot which prioritizes user engagement and implements continuous learning and self-improvement.

Objectives

1. **Enhancing User Interface (Software)**
2. **Continuous Learning and Self-Improvement Algorithm (Software and Hardware)**

ENHANCING USER INTERFACE (SOFTWARE)

Objective 1

User Interface Redesign Tools:

1. **Graphic design software** (e.g., Adobe XD, Sketch) for creating a visually appealing and user-friendly interface.
2. .**Front-End Development Framework:** Web or application development framework (e.g., Django, Flask for web-based UI) in Python for implementing the redesigned user interface.
3. **Responsive Design Libraries:** Libraries like Bootstrap or MaterializeCSS to ensure the user interface adapts to different screen sizes and devices.

Usability Testing Tools:

1. **Python testing frameworks (e.g., Pytest)** and usability testing tools for evaluating the redesigned user interface.
2. **User Documentation and Training Materials:** Python-based documentation tools (e.g., Sphinx) for creating user guides, tutorials, and help documentation.

Continuous Learning and Self-Improvement Algorithm (Software and Hardware)

Objective 2:

1	Reinforcement Learning Framework: Python-based reinforcement learning libraries (e.g., TensorFlow, PyTorch) for developing and training the self-improvement algorithm.
2	Data Collection Infrastructure: Hardware components (e.g., sensors, cameras) for collecting gameplay data and Python libraries (e.g., Pandas) for data processing and storage.
3	Algorithm Development Environment: Python IDEs (e.g., PyCharm, Jupyter Notebook) for algorithm development, testing, and fine-tuning.
4	Machine Learning Libraries: Python libraries (e.g., scikit-learn) for data analysis and machine learning model training.
5	User Experience Integration Tools: Python frameworks (e.g., Django or Flask) for integrating self-improvement mechanisms into the user interface, including graphics libraries (e.g., Matplotlib) for data visualization.
6	Performance Monitoring and Analytics: Python libraries (e.g., NumPy, pandas) for tracking performance metrics and machine learning model evaluation.

LITERARY SURVEY

1. Autonomous chess-playing robotic arm using Raspberry PI

Authors: *R. Srivatsan, S. Badrinath and G. Lakshmi Sutha*
Published in: *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*
Year: *2020*
Link: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9262351*

Merits:
1.High Accuracy in Chess Piece Detection
2.Modular Design for Ease of Maintenance
Demerits:
1. Complexity of Robotic Arm Control
2. Computational Intensity

2. Title: Collaborative Robot System for Playing Chess

Authors: *P. Kołosowski, A. Wolniakowski and K. Miatliuk*
Published in: *2020 International Conference Mechatronic Systems and Materials (MSM)*
Year: *2020*
Link: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9202398*

Merits:
1. Integration of Chess Logic with Robotics
2. Flexible Game Modes
Demerits:
1. Chessboard and Piece Estimation Accuracy
2. Open-Loop Manipulator Control

3. Title: Wizard chess: An autonomous chess playing robot

Author: *S. Sarker*
Published in: *2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*
Year: *2015*
Link: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7443971*

Merits:
1.Simplicity and Cost-Effectiveness
2.Accessibility and Educational Value
Demerits:
1.Precision and Reliability
2.Dependence on External Computer

4. Title: Chess piece movement detection and tracking, a vision system framework for autonomous chess playing robot

Authors: *D. A. Christie, T. M. Kusuma and P. Musa*
Published in: *2017 Second International Conference on Informatics and Computing (ICIC)*
Year: *2017*
Link: *https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8280621*

Merits:
1. Robust Chessboard Detection
2. Chess Piece Movement Detection
Demerits:
1. Dependence on Lighting Conditions
2. Complexity and Accuracy

5. Title: Implementation of an autonomous chess playing industrial robot

Authors: *J. Golz and R. Biesenbach*
Published in: *2015 16th International Conference on Research and Education in Mechatronics (REM)*
Year: *2015*
Link:*https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7380373*

Merits:

1. *Innovative Integration*
2. *Effective Technology Fusion*

Demerits:

1.Limited Real-World Testing
2.Limited User Experience Discussion

Methodology:

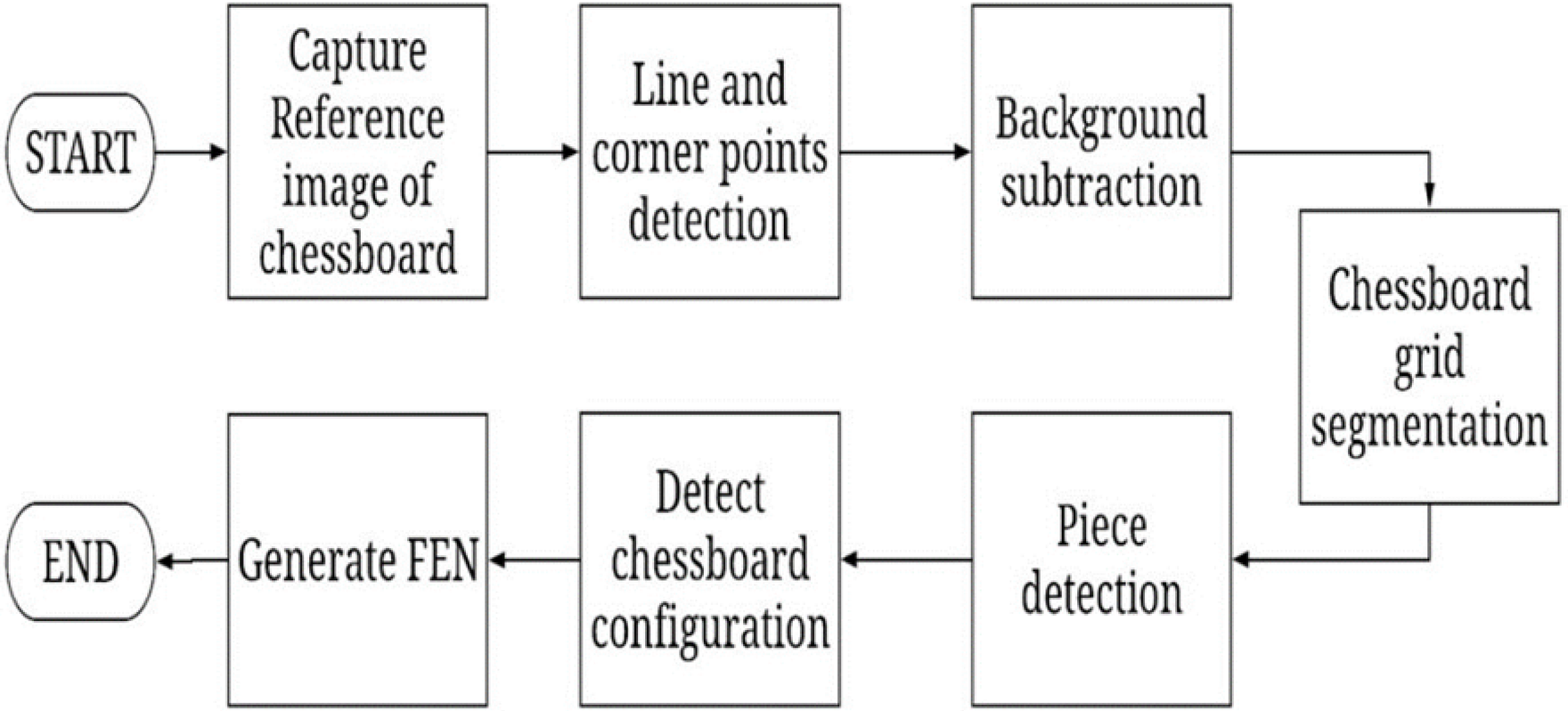
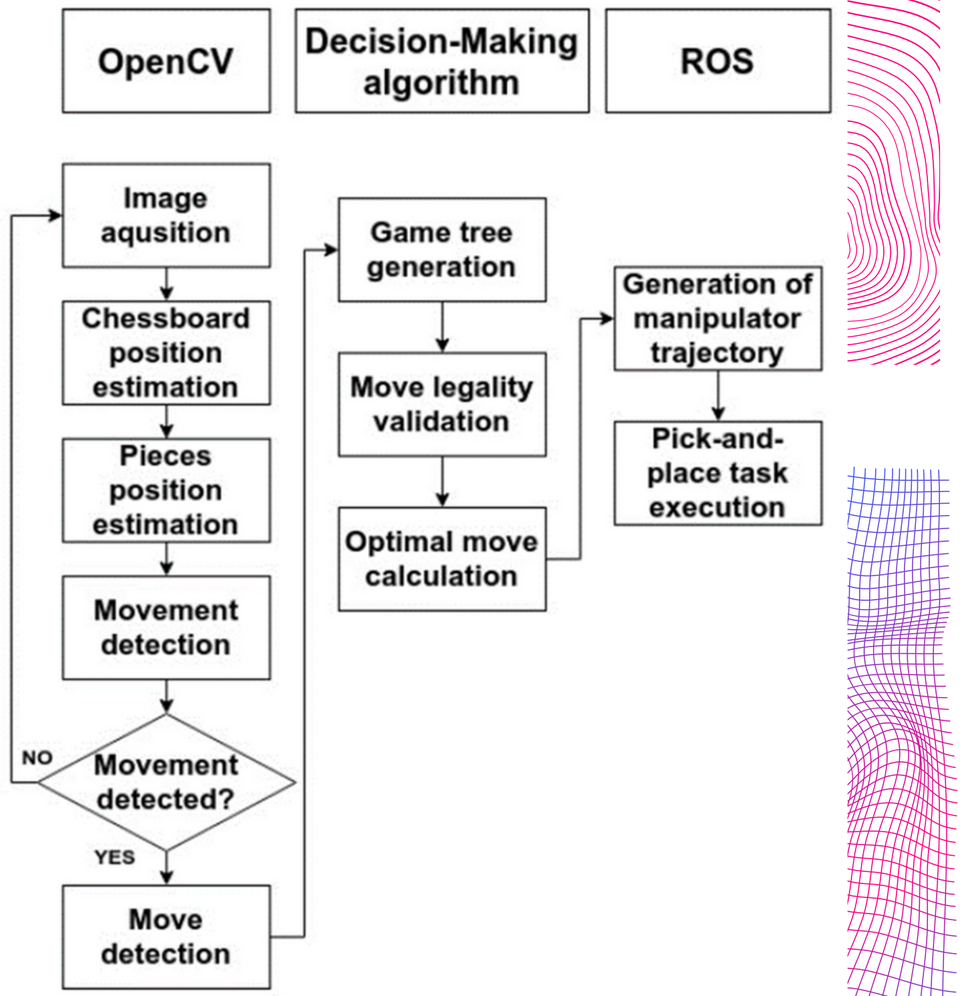


Image processing technique

Workflow of the system:

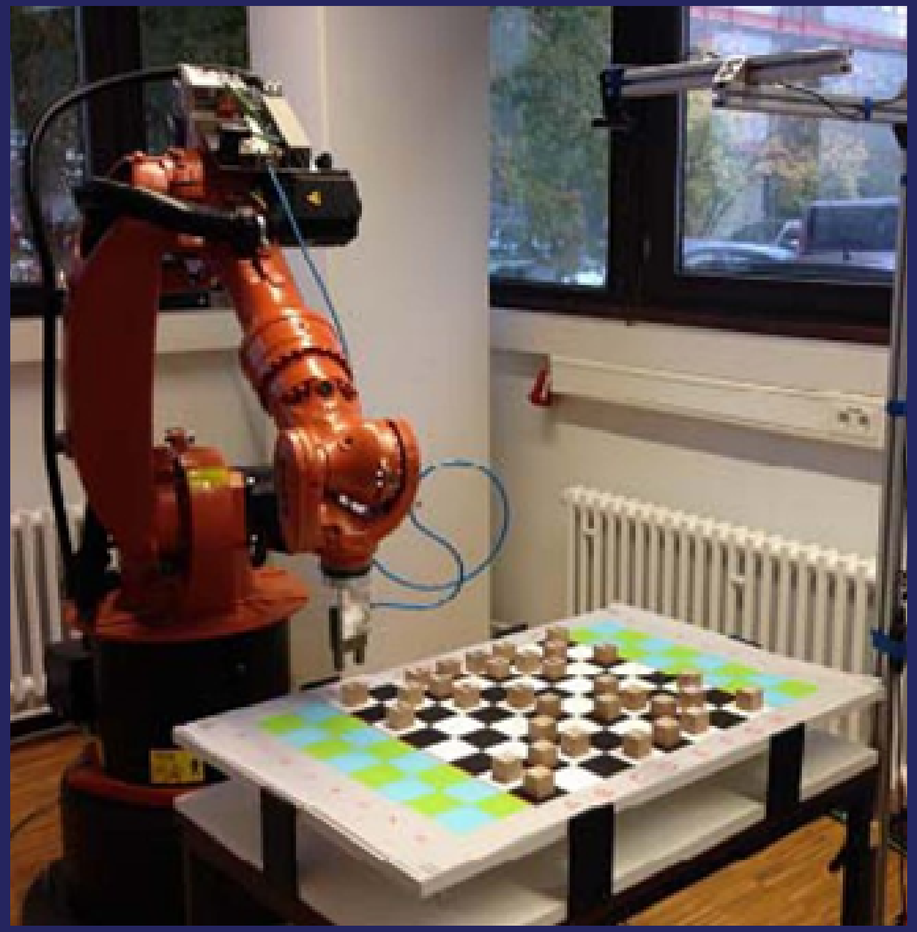
Line Intersection:
Chess cell can be determined from the intersection between the lines. Intersection of two lines in xy plane, where lines L1 represented as two points in space (x1, y1), (x2, y2), the intersection point P between line L1 and L2 can be defined using determinants. Determinants can be written out as equation:

$$(\mathrm{P}_x,\mathrm{P}_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right) \left(\frac{(x_1y_2 - y_1x_2)(y_3y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$



From 18 lines detected, 9×9 intersection exist, 81 points in total.

INNOVATION STATEMENTS

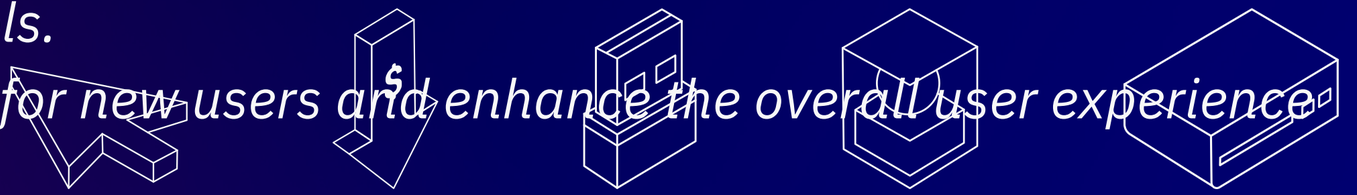


1. Usability:

Elaboration: *In the context of enhancing the usability of chess-playing robots, the incremental change focuses on making gradual improvements to the user interface (UI).*
This change recognizes that chess robots have traditionally been functional but may not always provide the most user-friendly experience.

Why It Matters: *Usability is a critical factor in the adoption and enjoyment of any technology. By incrementally improving the usability of chess-playing robots, we aim to make them more approachable to users of all skill levels,*
including beginners who may be less familiar with chess or robotics.

Approach: *The redesign of the UI will prioritize intuitive navigation,*
clear visual cues, and user-friendly controls.
The goal is to minimize the learning curve for new users and enhance the overall user experience.



2. Continuous Learning and Self-Improvement: Fine-Tuned Learning (Hardware and Software):

Why It Matters: *While the existing learning algorithm adapts based on historical data, this incremental change introduces a real-time fine-tuning process that continuously refines the robot's strategies based on the latest experiences.*

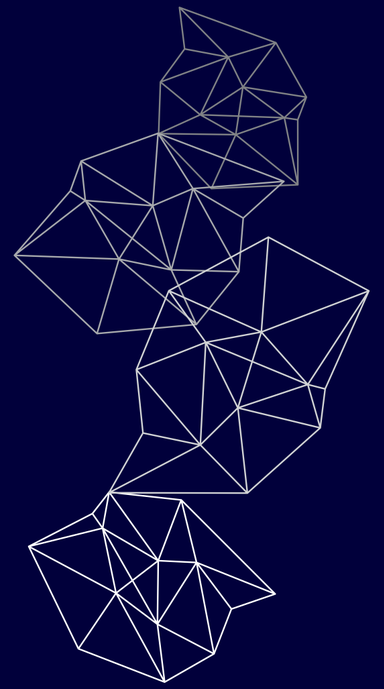
Hardware and Software Integration: *Ensure that the hardware sensors, data processing unit, and software reinforcement learning framework are designed to accommodate real-time data processing and adjustments.*

Approach:

1. Real-Time Data Processing: Develop algorithms for real-time data processing from sensors during gameplay, enabling immediate analysis of moves and performance.

2. Fine-Tuning Mechanism Integration: Integrate a fine-tuning mechanism within the reinforcement learning framework to make incremental adjustments based on real-time data insights.

Code and simulation



Video cam with raspberri pi

Code:

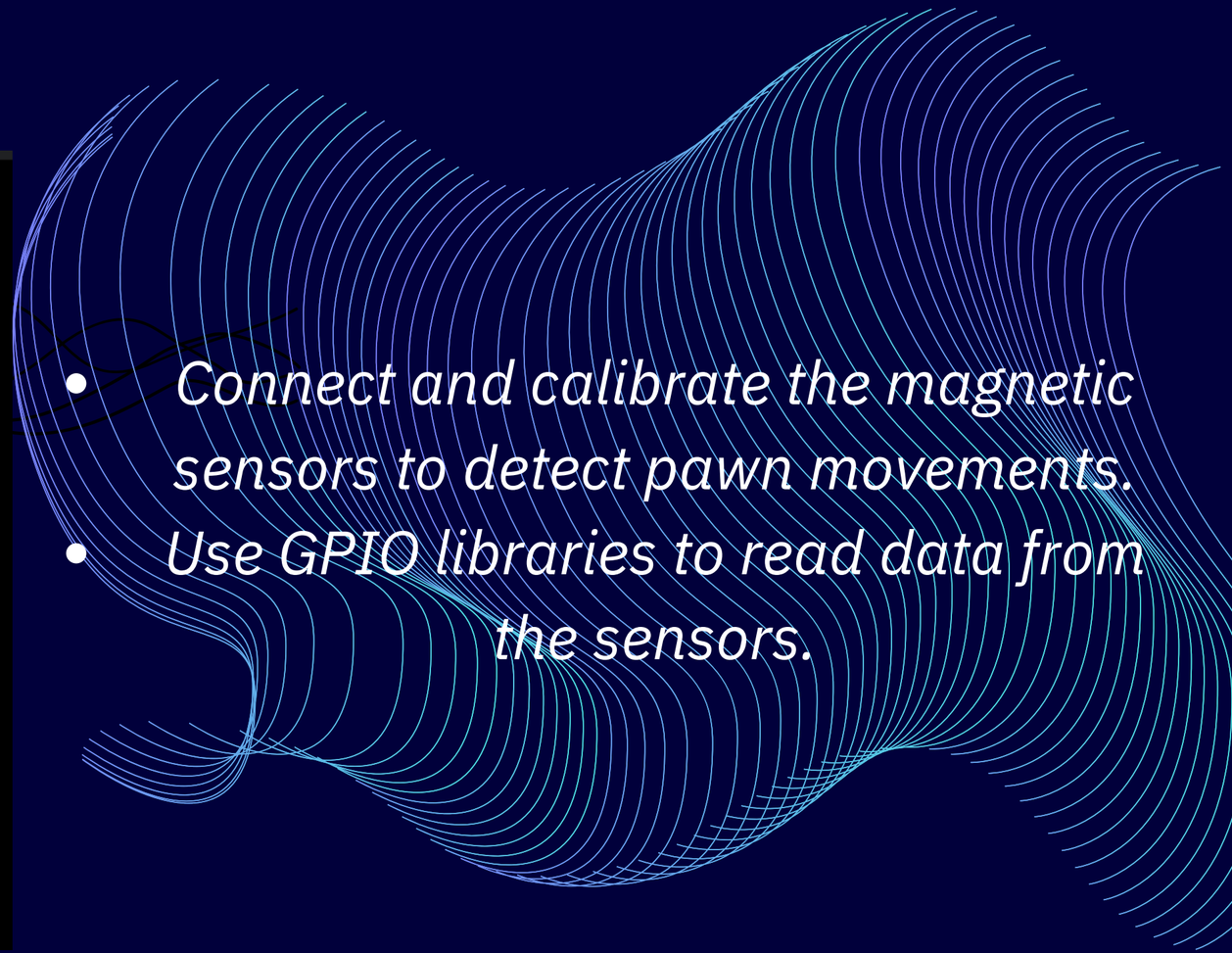
```python
import cv2

# Initialize the camera
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()

    # Process the frame here

    cv2.imshow("Chessboard Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Use OpenCV to capture video frames from the webcam.
Implement chessboard detection to identify the current state of the chessboard.

Magnetic Sensor Integration:

Code:

```
import RPi.GPIO as GPIO

sensor_pin = 17  # Example GPIO pin

GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN)

while True:
    sensor_value = GPIO.input(sensor_pin)

    # Process sensor data here

GPIO.cleanup()
```



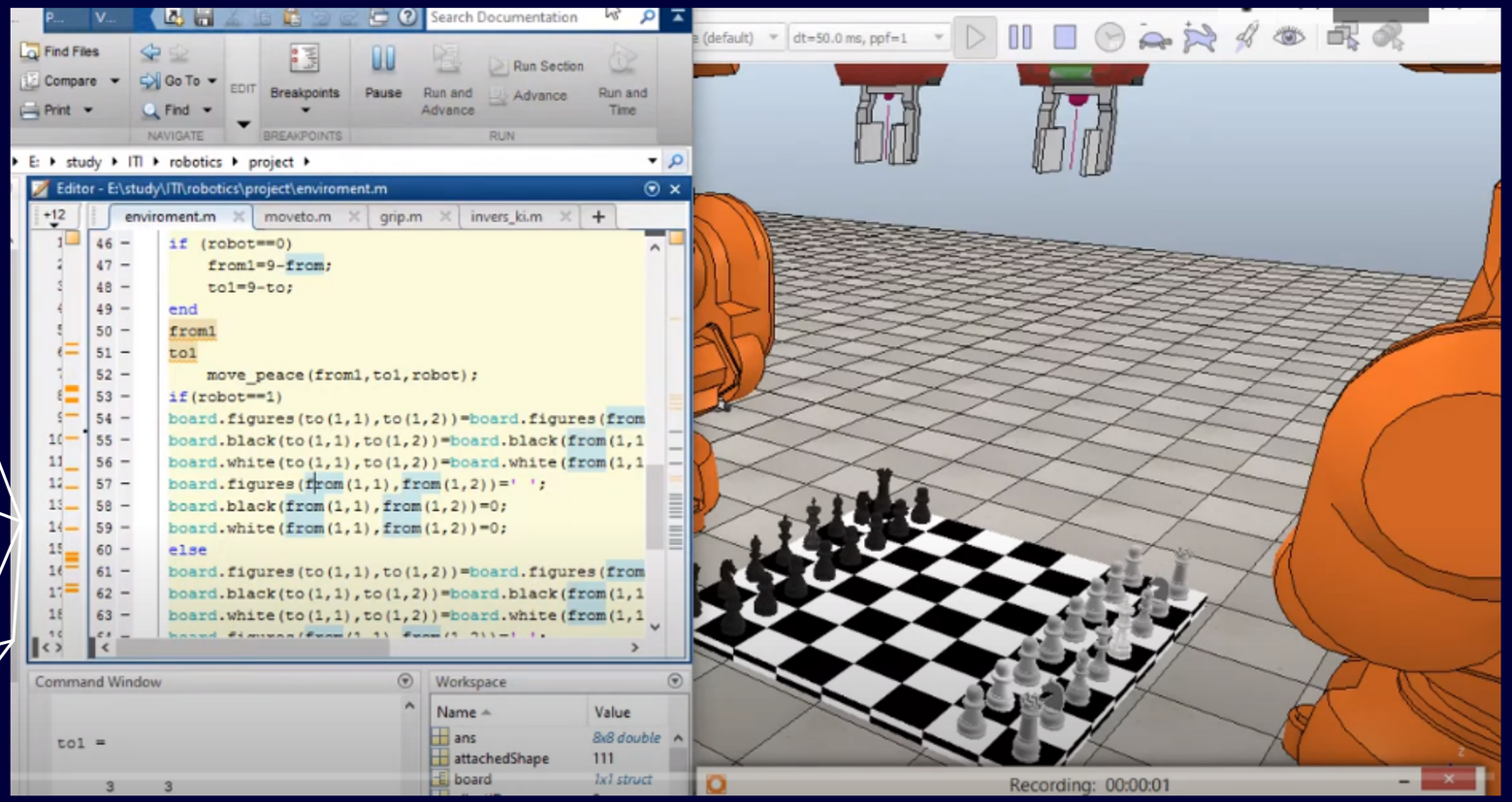
- *Connect and calibrate the magnetic sensors to detect pawn movements.*
- *Use GPIO libraries to read data from the sensors.*

Simulation using VREP:

1. ***Environment Creation**: V-REP offers a user-friendly interface to build 3D virtual environments by adding objects, robots, sensors, and various components using drag-and-drop functionality.*
2. ***Robotic System Modeling**: Users can construct and simulate robotic systems, including robot arms, drones, or mobile robots, by assembling components like joints, sensors, and actuators within the scene.*
3. ***Physics Simulation**: V-REP accurately simulates physics, accounting for factors like collision detection, friction, and gravity, replicating realistic interactions within the virtual environment.*

4. ***Scripting and Control:*** *V-REP allows programming of robots using Lua scripting or through remote APIs (such as Python or C/C++), enabling control over robot movements, behaviors, and interactions.*

5. ***Sensor Simulation:*** *It supports simulation of various sensors like cameras, LiDAR, or proximity sensors, allowing users to configure and simulate sensor data output crucial for robot perception and decision-making.*



THANK YOU