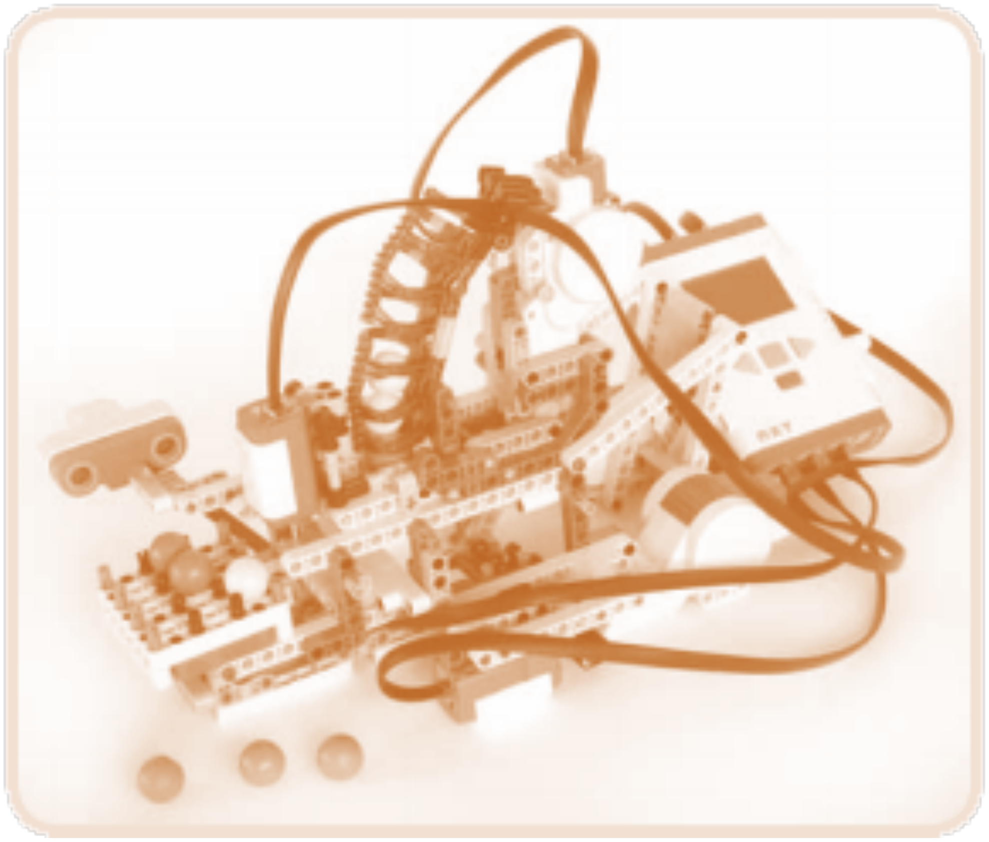
Tic-Tac-Toe Bot



The only winning move is to play, perfectly, waiting for your opponent to make a mistake.

~ XKCD

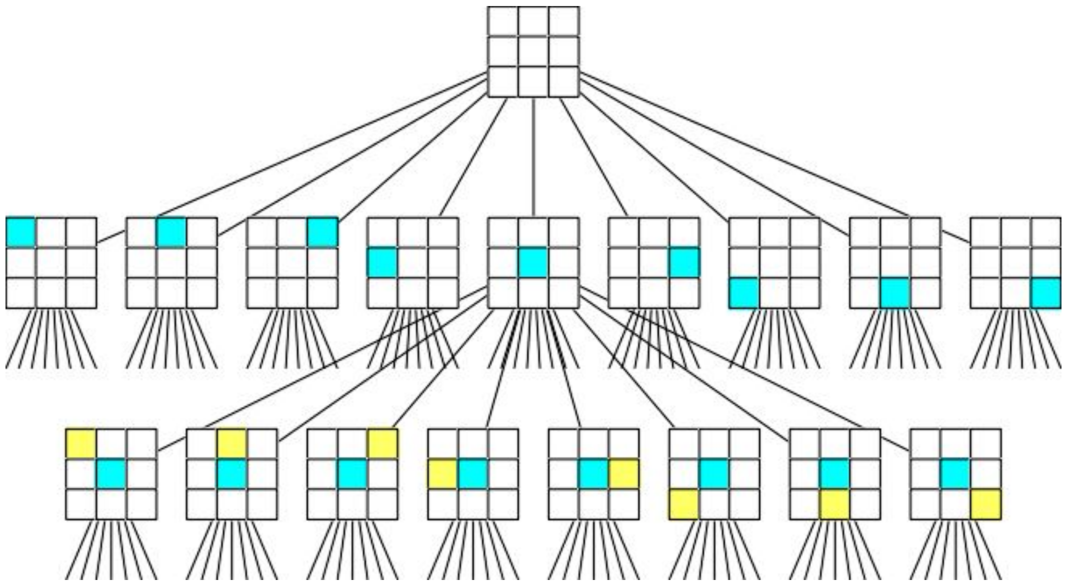
The GETigers engineering firm has been contacted by a robotics company doing research in artificial intelligence. However, there is competition for the contract. In order to choose the best of the bidders, the robotics company has devised a test: program a tic-tac-toe playing robot.

Your design team must implement the logic for the game in Matlab. You will be supplied a robot using a Lego Mindstorms kit to integrate your program. More details on the scoring criteria and requirements for this project are outlined below.

Background

Artificial intelligence is often defined as the design of agents that perceive their environment and act to maximize their chances of succeeding in their goals. Artificial intelligence (AI) has many applications, such as medical diagnosis, computer vision, stock trading, robotics, autonomous vehicles, etc. One place nearly everyone has seen artificial intelligence at work is in video games. Whether in checkers or Starcraft, video games almost always feature a computer opponent. In order for this to work, the program must have some way of evaluating its current position and determining the move that will most improve it.

One common method for artificial intelligence in games is the use of a game tree. In a game tree, each node of the tree represents a state of the game. The edges between nodes represent possible moves. Often, a game AI will calculate a partial game tree representing the possible game states for the next few moves.



Unlike large games such as chess, the game tree for tic-tac-toe is small (26,830 possible states that end the game). Thus, a computer can calculate the full game tree instead of a partial one, allowing the computer to play a perfect game every time.

Specifications: Part I

- Develop logic for the computer to select its best move in a game of tic-tac-toe.
- Implement a program that allows a user to play a game against the computer.

Game Logic Design — Wednesday April 4 – Friday April 13

In this part, you will design the artificial intelligence module to allow the computer to play the game. Note that since tic-tac-toe is a fully determined game, a proper implementation will mean that the computer never loses.

- The board should be represented with a 3x3 matrix. Each index in the matrix will store a character: 'X' or 'O' for a space occupied by a player, or '-' for an empty space.
- A move should be represented with a 1x2 matrix that contains the row and column of the move.
- Your game logic should be implemented as a function that accepts a board data structure as its only argument and returns a move data structure.

Interactive Game Program — due Friday April 13 at midnight

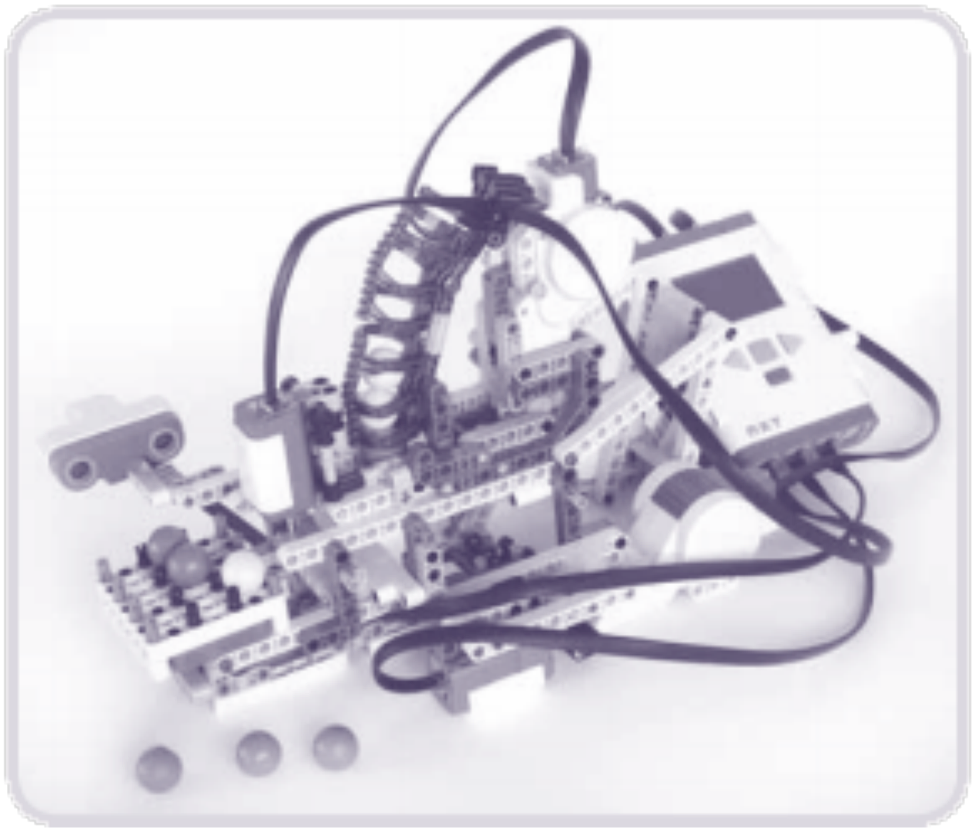
The artificial intelligence that allows the computer to select a move is only part of the game. You must incorporate this into a program that allows a human to play against the computer.

- ❑ The program should allow the user to enter a name, and then call the user by the name entered throughout the program.
- ❑ At each turn, the program should display whose move it is (human or computer). After a move is made, the program should display the current state of the board.
- ❑ The human should enter moves as a row number, then one or more spaces, and then a column number. There must be one prompt for the whole move, not separate prompts for row and column. Valid row/column numbers are in the range 1-3. You must include error checking to re-prompt the user if they enter an invalid move.
- ❑ When the game is first started, a menu should ask whether the human or computer plays first. There should also be an option to have the computer play a game against itself.
- ❑ Your game function should return the final state of the board and state a winner (human, computer, or tie).

It is suggested, for efficiency, you examine ways to use functions throughout your code. Any other supporting functions you write must be included in the final submission. All code files should include a standard header containing your group number, the names of all group members, and a high level overview of what the code in the file does.

You must also submit a 1-page memo explaining the method you used to determine the computer's best move. This should be compared and contrasted to any alternatives that you came up with or found in your research, and include citations for any sources you used.

Tic-Tac-Toe Bot



Video games are bad for you? That's what they said about rock 'n' roll.
~ Shigeru Miyamoto (Nintendo)

The GETigers engineering firm has been contacted by a robotics company doing research in artificial intelligence. However, there is competition for the contract. In order to choose the best of the bidders, the robotics company has devised a test: program a tic-tac-toe playing robot.

Your design team must implement the logic for the game in Matlab. You will be supplied a robot using a Lego Mindstorms kit to integrate your program. More details on the scoring criteria and requirements for this project are outlined below.

Specifications: Part II

Playing a game of tic-tac-toe inside MATLAB is fun, but being able to play a game against the computer with a real board and real pieces is much better. In this part of the project, your team will implement some utility functions to allow the robot to manipulate a tic-tac-toe board. You will then integrate your game logic from Part I with the robot to produce the final submission.

Robot Interface Workdays — Monday April 16 – Monday April 23

Your project will interface with a standard robot with MATLAB using the RWTH Mindstorms for MATLAB toolbox. For this project, you will need to implement three functions.

- The first function should accept a move data structure (as described in Part I) and physically place a marker on that space on the board. The multi-colored balls included in the Mindstorms kits will serve as the markers.
- The second function should scan the board using the color sensor and output a board data structure (as described in Part I).
- The third function is an initialization routine. This should reset all moving parts to their desired starting positions. This function should also accept two arguments defining which color balls correspond to 'X's and 'O's for the game.

Documentation for the RWTH toolbox is available on their website. The lab manual for the ECE section of ENGR 190 may also be a useful resource for you, especially for installing and configuring the RWTH toolbox.

http://www.mindstorms.rwth-aachen.de/trac/wiki/Documentation

http://www.clemson.edu/ces/departments/ece/academics/undergrad/mindstormslab.html

Robot Interface Program — due Monday April 23 at midnight

Your deliverable for this section should contain the following:

- ❑ MATLAB code for all three required functions, and any additional supporting functions or code you write must be included as well. All code files should include a standard header containing your group number, the names of all group members, and a high level overview of what the code in the file does.

- ❑ Videos of your robot scanning the board and placing a marker. These should be in Quicktime (.mov), MPEG (.mp4), AVI (.avi) format.

For this portion of the project, you will have access to the robots at the following times. Please plan accordingly! You will NOT be permitted access to a robot at times other than those listed below.

Monday April 16	*regular class period*	*7:00 – 9:00 pm for sections 41, 42, 45, 49*
Tuesday April 17	*11:00 am – 4:30 pm*	*7:00 – 9:00 pm for sections 40, 43, 44, 47, 51*
Wednesday April 18	*regular class period*	*7:00 – 9:00 pm for sections 41, 42, 45, 49*
Thursday April 19	*11:00 am – 4:30 pm*	*7:00 – 9:00 pm for sections 40, 43, 44, 47, 51*
Friday April 20	*regular class period*	
Sunday April 22		*7:00 – 9:00 pm for sections 41, 42, 45, 49*
Monday April 23	*regular class period*	*7:00 – 9:00 pm for sections 40, 43, 44, 47, 51*

Specifications: Part III

Integration Workdays — Monday April 23 – Friday April 27

At this stage, you have completed code to accept input from players, decide moves for the computer, and make moves using the robot. Now it is time to tie all that together.

Your final program should be much like the game function implemented in Part I. It should repeatedly prompt the player for a move, make a move for the computer, and display the board state. However, now instead of the user entering a move on the keyboard, they must be able to enter moves simply placing one of their markers on the board.

To detect when the user has made a move, your program should make use of the ultrasonic sensor mounted next to the board. Instead of the computer making a move by simply changing a value in the board matrix, it must physically place a marker on the board using the robot.

Your program should still display in MATLAB whose turn it is and the state of the board after each turn. Also, as in part one, when the program is first run, it should prompt for who plays first and have an option to allow the computer to play against itself.

For this portion of the project, you will have access to the robots at the following times. Please plan accordingly! You will NOT be permitted access to a robot at times other than those listed below.

Monday April 23	*regular class period*	*7:00 – 9:00 pm for sections 40, 43, 44, 47, 51*
Tuesday April 17	*11:00 am – 4:30 pm*	*7:00 – 9:00 pm for sections 41, 42, 45, 49*
Wednesday April 18	*regular class period*	*7:00 – 9:00 pm for sections 40, 43, 44, 47, 51*
Thursday April 19	*11:00 am – 4:30 pm*	*7:00 – 9:00 pm for sections 41, 42, 45, 49*
Friday April 20	*regular class period*	

Integration Results — due Friday April 27 at midnight

Program: Your final submission must also include any and all MATLAB code you wrote to implement the project. All code files should include a standard header containing your group number, the names of all group members, and a high level overview of what the code in the file does.

Presentation: Each team must submit a powerpoint presentation, including the following components:

- ❑ Discussion of the methods used to determine the computer's best move for a given board state, including a set of examples. Justification of why you selected your method over others you may have read about.
- ❑ Discussion of the methods used to scan the current board state and correctly place a marker for a given move.
- ❑ Videos of one of your group playing a game of tic-tac-toe against your robot and of the robot playing against itself.
- ❑ Citations for any sources you may have used.

A PowerPoint presentation using voice-over should be developed to address the requirements given above. Information on developing a presentation is given in your textbook in Section 4.1, and the accompanying online slides discuss voice-over. The presentation will be graded on neatness, organization, spelling/grammar/mechanics, and strength of conclusions.