Department of Electrical Engineering and Computer Science

Howard University

Washington, DC 20059

EECE404: Senior Design II

Spring 2023

EECE404: Senior Design II

Terminator

By:

Omaretsoguwa Atsagbede

Oshione Adams

Bella Kuete

Amiyah Brown

Faculty Advisor: Dr. Charles Kim

Date Assigned: 4/18/2023 Date Submitted: 4/25/2023

Summary

The project aims to reduce children's excessive phone usage by designing an interactive robot that plays tic-tac-toe. The final design incorporates a Mega Pi board, two motors to move a pen holder in the x and y planes, a ribbon system, and USB connectivity. The robot is capable of playing on a 10x10 grid and has been tested thoroughly using different pen holders and various game moves to ensure reliability.

The team's development process involved multiple sprints, during which they focused on software development, hardware assembly, and refining the device. Challenges included creating a smart CPU function to counter player moves, finding a compatible pen holder, and training the CPU to make intelligent decisions using the Minimax Algorithm. The team addressed these issues by conducting extensive research and testing throughout the project timeline.

Overall, this interactive tic-tac-toe playing robot provides a stimulating alternative to screen time, promoting cognitive development and enhancing social skills. While the current design allows for play on a 10x10 grid, future iterations could expand to more complex games like chess. The project's success is due to rigorous testing and commitment to addressing challenges and minimizing failure risk.

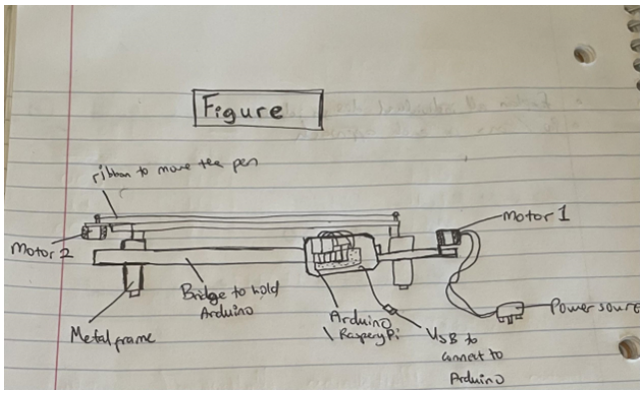
Problem Statement

Artificial Intelligence is the need of today's generation because children are always on their phone or don't have someone to play with, which led us to create a robot that plays tic-tac-toe, so that they can be productive and not be on their phones the whole day.

Design Requirement

Our Terminator project's design requirements have been carefully crafted to create a highly efficient and user-friendly device. Our team has designed the system to work with various AC input voltages, ranging from 100 to 240V, ensuring compatibility with multiple power sources. We've also focused on keeping the final product lightweight and easy to handle, achieving a net weight of around 3.87 kg and a gross weight of 4.475 kg. For programming, we'll be using Google Collab with Python, employing a brute force approach without depending on any in-built libraries. Our project includes 1 Arduino and 1 Laserbot for seamless control and operation, along with 2 motors for reliable movement and functionality. To refine our Tic-Tac-Toe strategy, we'll be exploring a wide array of online documents and resources. We've given careful thought to the device's physical dimensions, deciding on 535mm x 637mm x 184mm, and we aim to reach a maximum working speed of 200mm/s. This sleek and efficient design is intended to meet our project's goals while offering a delightful user experience.

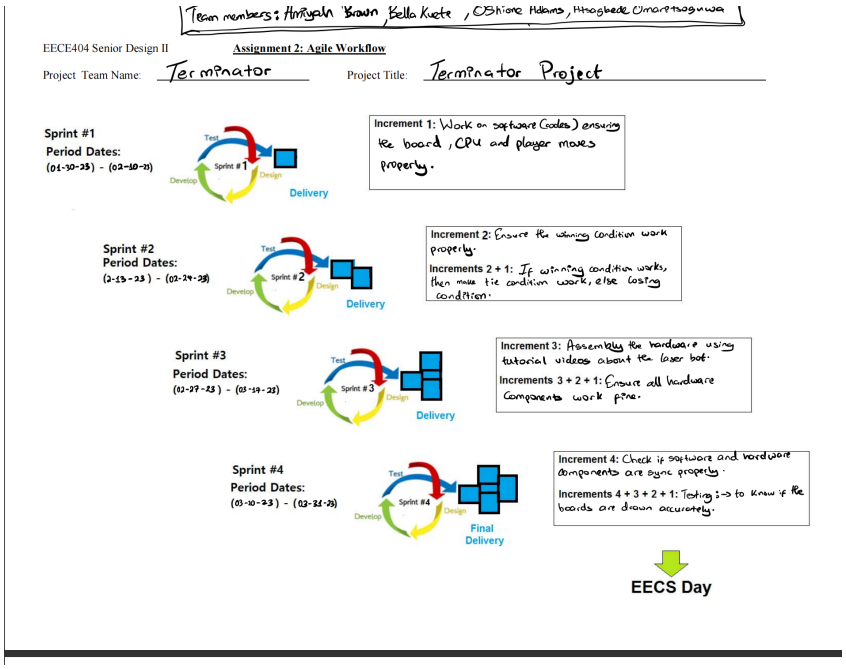
Solution Design



In the design depicted above, I've developed a device that incorporates various components, such as motors, a Mega Pi board, a USB port, a power source, metallic bridges, ribbons, and metal frames. The device uses two motors to control the pen's movement on both the x and y planes, with the pen holder effortlessly sliding along a ribbon. A reliable power source ensures efficient device operation, and the Mega Pi board's compatibility with both Arduino and Raspberry Pi makes it highly versatile and user-friendly. The device is supported by four robust steel bridges that hold the movable components and the Mega Pi board, while four metallic frames offer added stability. A USB port is integrated for easy connection of the Mega Pi board to a computer, enabling seamless code execution. However, some aspects of the design still need further exploration. One challenge we encountered was determining how to enable the device to draw diagonal movements, or the z-axis, on the tic-tac-toe board, considering the current design only permits movement along the x and y planes (horizontal and vertical planes). On a positive note, the Mega Pi integration streamlines the operational process, as it is compatible with both Raspberry Pi and Arduino platforms. Portability is somewhat limited, given the device's weight of 3.87 kg. Lastly, the device runs on an AC input voltage of 100-240V, reflecting an environmentally conscious approach. It's crucial that we thoroughly investigate and address these concerns to ensure the device operates at peak performance and delivers an outstanding user experience.

Project Implementation

Agile workflow:



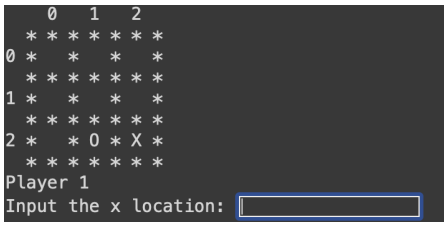
The project is divided into four sprints, each consisting of two increments, with specific weekly development tasks to be completed within each increment. In the first sprint, the team will focus on interfacing their code with the hardware, testing the overall system with both hardware and software, and ensuring that the board, CPU, and player moves properly. In the second sprint, the team will ensure that all winning conditions work, create a tie condition, and make a losing condition. The third sprint will involve assembling the hardware using tutorial videos and testing each component to ensure everything works fine. Finally, in the fourth sprint, the team will make sure that both the software and hardware components connect with each other properly, and will test for accurate values drawn by the hardware.

Project Implementation Process

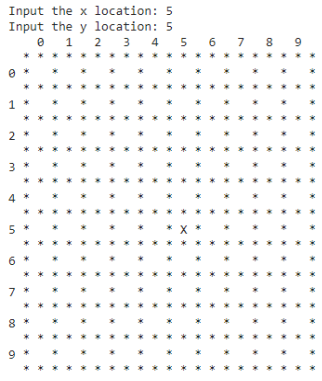
To build the device, the team will first assemble the frame, install the motors, and set up the ribbon system. They will then mount the Mega Pi board and connect the motors to the board. Once the hardware is in place, the team will power on the device and attach the pen holder. The next step is to implement USB connectivity and develop/upload the code to the device. After the code is uploaded, the team will modify the software to enable the device to play tic tac toe. This will involve programming the device to recognize player moves, make its own moves, and follow the rules of the game. The team will test and refine the system as needed to ensure proper functionality. Through this process, the team will have built a functional device capable of playing tic tac toe.

SPRINT #1 [01-30-23 to 02-10-23]

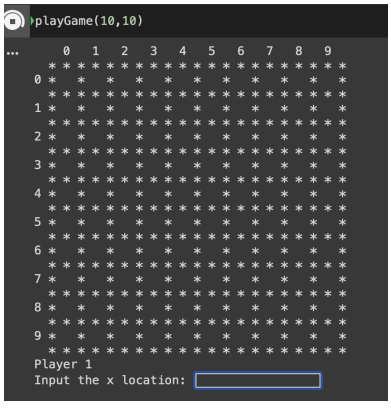
For Sprint #1, our main focus was to make sure our software for the CPU and the player movements work smoothly on a 10x10 game board. For our planned tasks, during week 1, we have done research to see different Tic Tac Toe boards and to find a 10x10 game board version. From there, we created the game board design and then we had to find requirements necessary for the winning, losing and drawing conditions. During week 2, we finalize the movements for each piece on the tic tac toe board for our code. We worked on the movements by the CPU, ensuring it makes proper countering moves. Finally, we had to ensure board displays at the right positions for the X and O movements. What went well during this sprint was that we were able to make the codes for the player 1 vs player 2 and make the codes for player 1 vs CPU, which became our main code. Some of the tasks that were not completed were completing the moves for the computer to counter the moves properly like if it is a real game. We also didn't apply machine learning for the computer to move on its own. We didn't have enough time to teach the computer to properly block the moves coming from Player 1 properly.



3x3 Board for Player1 vs Player2

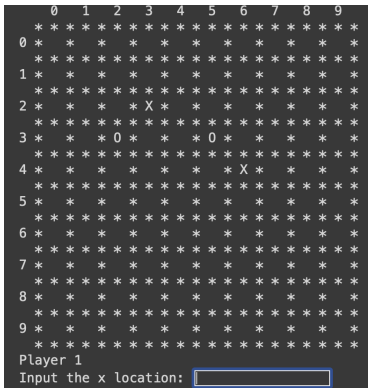


10 x 10 Board for Player1 vs Player2



10 x 10 Board for Player1 vs CPU

I attached three different tic-tac-toe boards, to show the progress from 3x3 player 1 vs player 2, to now 10x10 player 1 vs computer.



Here, on this 10 x 10 Board Player 1 vs The Computer, it shows the the movements of 'X' and 'O' elements

Overall, in order for us to solve the issues, we had to research more and fully understand the techniques in order to block the player 1 moves. We also ran and tested more codes to teach the CPU techniques to block the player 1 moves. In order to reduce failure we must test the codes more often and test the code using very unique and different test cases.

SPRINT #2 [02-13-23 to 02-24-23]

For Sprint #2, our main focus was to ensure all the conditions in the code, such as the winning, losing, tie were working properly. For our planned tasks, during week 1, we worked on the winning conditions and focused on building rows, columns, right & left diagonals. During week 2, we worked on the losing and draw conditions; if nobody wins, that would be because the board is full which will result in a draw.

```
def checkWin(tttboard,x,y,tar,ch):
        if checkrows(tttboard,x,y,tar,ch):
          return True
        elif checkcolumns(tttboard,x,y,tar,ch):
          return True
        elif checkrightdiag(tttboard,x,y,tar,ch):
          return True
        elif checkleftdiag(tttboard,x,y,tar,ch):
          return True
        else:
          return False
```

Here is our code, which represents the different ways to win.

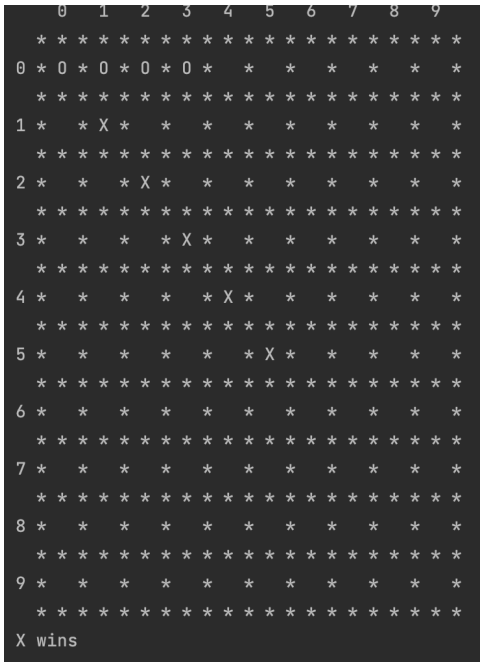
What went well during this sprint was that winning, losing, and draw conditions worked. The main tasks that were not completed was the computer function and because of that, the moves were not moving as smartly as we wanted it to.

```
if currPlayer:
    tttboard[x][y] = "X"
    if checkWin(tttboard, x, y, thresh, "X"): # check if X has won
        drawBoard(tttboard)
        won = "X wins" # won is no longer None. While loop breaks
        continue
else:
    tttboard[x][y] = "O"
    if checkWin(tttboard, x, y, thresh, "O"): # check if Y has won
        drawBoard(tttboard)
        won = "O wins" # won is no longer None. While loop breaks
        continue

# if nobody has won
drawBoard(tttboard) # show the updated board
plays += 1 # increase the number of those who have played
currPlayer = not currPlayer  #change the player to the CPU

# if people have played n^2 times, the whole board is filled.
# it's a draw
if plays == n**2:
    won = "DRAW"
```

Here, it is our code where if X has won the match, it will print out X wins. If O wins, it will print out O wins. If nobody wins, then it will end in a draw.



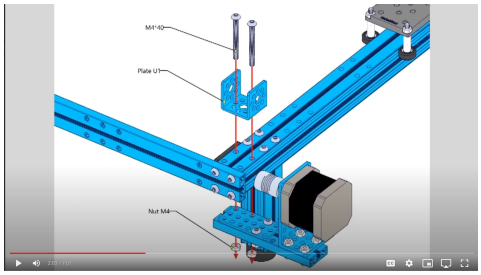
Here, this is an example of “X” winning diagonally, showing how the winning condition works.

Overall, in order for us to solve the issues, we had to research on how to train the CPU function using the minimax algorithm. Then, we must run the code numerous times with different test cases in order to pinpoint errors in the code. For us to reduce failure risk, we had to test the codes at different increments to make sure all of the conditions worked properly, test the code at different IDE’s and to do multiple test at the end to verify that everything works.

SPRINT #3 [02-27-23 to 03-17-23]

For Sprint #3, our main focus was to assemble the hardware using tutorial videos about the laser bot.

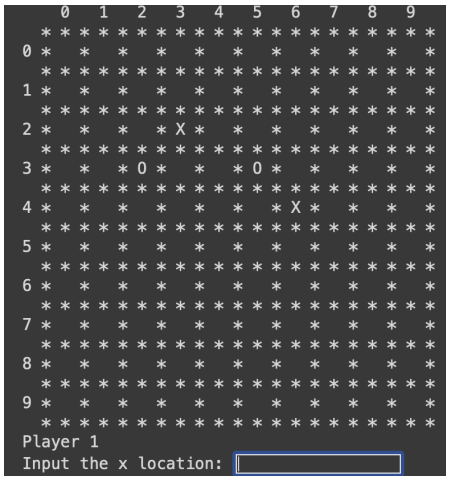
For our planned tasks, during week 1, we took the time to watch and understand the hardware tutorial and then work up to halfway through the assembling tutorial. During week 2, we finished assembling the hardware all together and made sure all of the components of the hardware works and operates as it should.



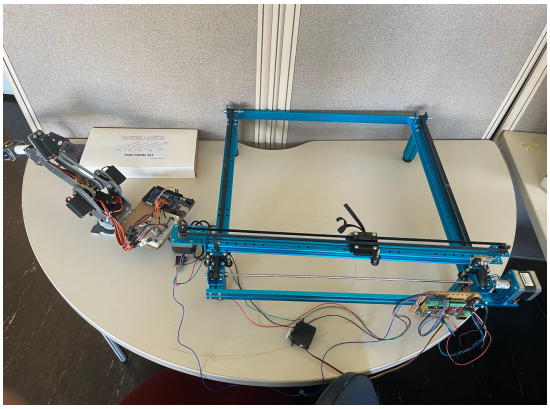
Tutorial Video - link to tutorial video

What went well during this sprint was that we were able to get both motors to work fine, since previously, there was little voltage coming from the power source from one of the motors. The remaining of our hardware components was fine and we were able to have the computer function work more successfully and make smarter move from Sprint #2 works properly. The main tasks that were not completed

successfully was removing the laserbot lasers with a pen holder, since we weren't able to find a pen holder that was compatible with the laserbot.



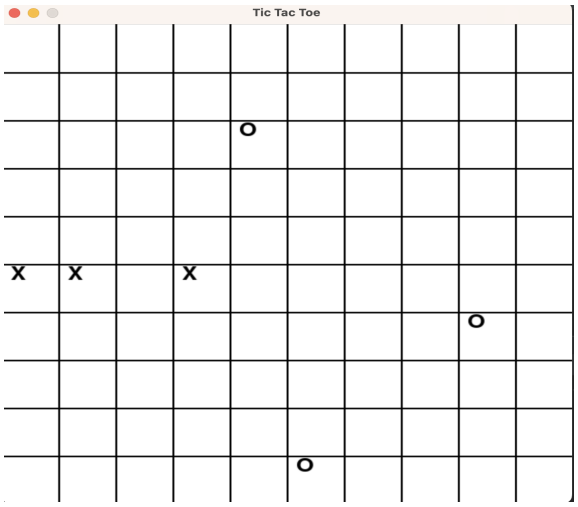
10 x 10 Board Player 1 vs CPU from Sprint #2



Picture of the laserbot from LKD

Overall, in order for us to solve the issues, we had to make sure to find a compatible pen holder and attach it to draw out the board from the code. In order for us to reduce the failure risk, we had to test different pen holders and we tested different play moves on the 10 x 10 board.

SPRINT #4 [03-20-23 to 04-11-23]



In our recent sprint, we found out that the hardware just couldn't handle our code properly because of the way the 10x10 board grid was set up – it was in a star format. To fix this, we had to tweak the code so that it used straight lines around the board instead. You can see the change we made in the picture above.

During this sprint, we also focused on optimizing the CPU function to not only counteract the player's moves, but also to proactively seek opportunities for victory. One of the key strategies we implemented was to have the CPU prioritize keeping the board's center unoccupied, allowing it to take advantage of

that position when launching an offensive against the player. This strategy can also be seen in the image above for spring #4.

Conclusion

In conclusion, the primary goal of this project was to help children reduce excessive phone usage by introducing an interactive tic-tac-toe playing robot. Key design components included the Mega Pi board, motors, and a pen holder, which enabled gameplay on a 10x10 grid. Future iterations of the project could expand the robot's capabilities to more complex games like chess. The team conducted thorough testing, including assessing different pen holders and game move strategies, to ensure the device's reliability and minimize the risk of failure. The resulting tic-tac-toe robot offers numerous benefits for children, such as providing a stimulating alternative to screen time, promoting cognitive development, enhancing social skills, and presenting an engaging way to play a classic game.

References

- [1] S. Kelly, Python, PyGame, and Raspberry Pi Game Development. New York: Apress, 2019.
- [2] T. Wu and C. Yu, "Tic-tac-toe prediction based on machine learning methods," in 2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Wuhan, China, 2022, pp. 397-402, doi: 10.1109/AEMCSE55572.2022.00085.
- [3] D. B. Fogel, "Using evolutionary programming to create neural networks that are capable of playing tic-tac-toe," in IEEE International Conference on Neural Networks, San Francisco, CA, USA, 1993, pp. 875-880 vol.2, doi: 10.1109/ICNN.1993.298673.
- [4] Agmed, tech_support, and System, "Laserbot laser Makeblock," Makeblock Forum, Aug. 06, 2020. [Online]. Available: https://forum.makeblock.com/t/laserbot-laser-upgrade/17029.
- [5] Real Python, "Build a tic-tac-toe game engine with an AI player in python," Real Python, Feb. 24, 2023. [Online]. Available: https://realpython.com/tic-tac-toe-ai-python/.
- [6] "About MegaPi – makeblock help center." [Online]. Available: https://support.makeblock.com/hc/en-us/articles/4412894483095-About-MegaPi.