Perceptron Guide

Students Participants:

Chloe A.

Theodore M.

Mateo H.

Team: 4330R

Location: Miami, FL

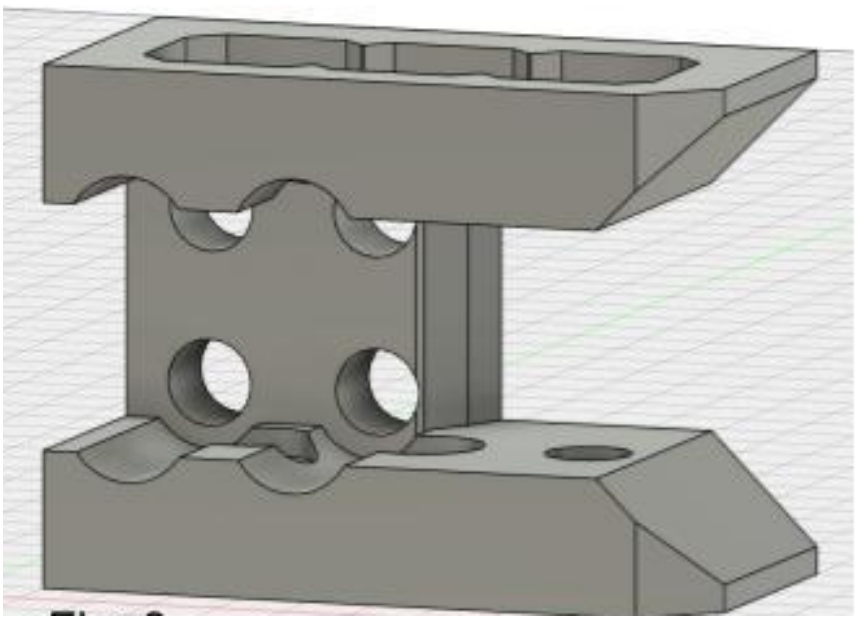


Fig. 1

Our inspiration for this project came from the notion of creating a robot that could accomplish a feat humans could not. After a team discussion, we arrived at a robot that could perceive shapes -- albeit basic ones -- in the dark. We planned to create a learning set by showing the robot the shapes and specifying which array belonged to which shape. Then, we would conduct a second trial of showing the robot the shapes and observing the accuracy with which SID identified the shapes. SID stands for Seeing In Darkness; Figure 1 depicts SID. The integral component of SID is the Perceptron Guide, which we designed using Autodesk Fusion 360 (version used Autodesk Fusion 360 © 2020, 2.0.11415).

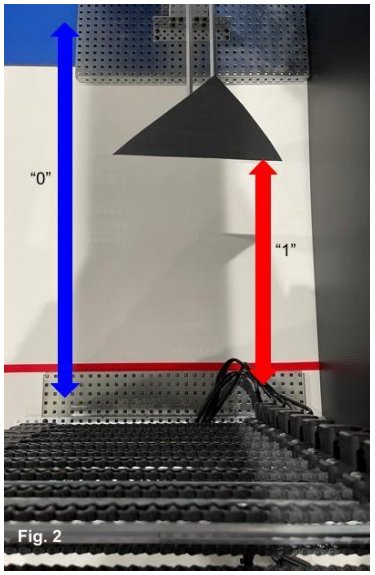

Fig. 2

To succeed with SID, we aligned ten distance sensors horizontally, attached to a chain (Figure 1). The distance sensors move small distances to the right, measuring the distance at intervals from their location on SID to a basic shape (either a circle, triangle, or square). Then, SID's program converts these distances to 1s and 0s. After putting all the 1s and 0s together in an array, a circle, triangle, or square is identified. Once the learning phase is complete, we will test the SID's accuracy.

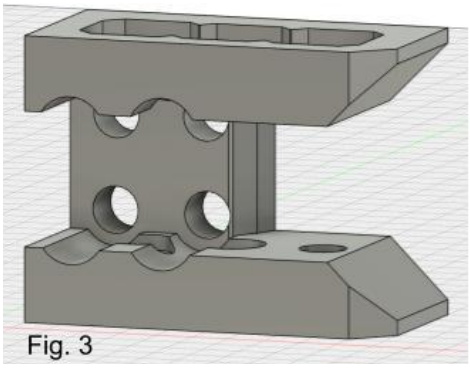
Fig. 3

To take these readings the sensor will have to move laterally precisely while retaining the orientation of the distance sensor. We devised a piece (Fig. 3) measuring 25 mm by 49 mm by 39 mm that houses a six tooth sprocket, supports a VEX V5 motor, and accepts two VEX square shafts. For this discussion, we will call this piece the Perceptron Guide. The VEX shafts would come out from the sides of the Perceptron Guide and, using the chain, move a shuttle housing a sensor from side to side. (Fig. 4) The sensor affixes to the shuttle and the chain to be moved. This apparatus also has the advantage of allowing one motor to move many sensors both in SID and a mobile robot.

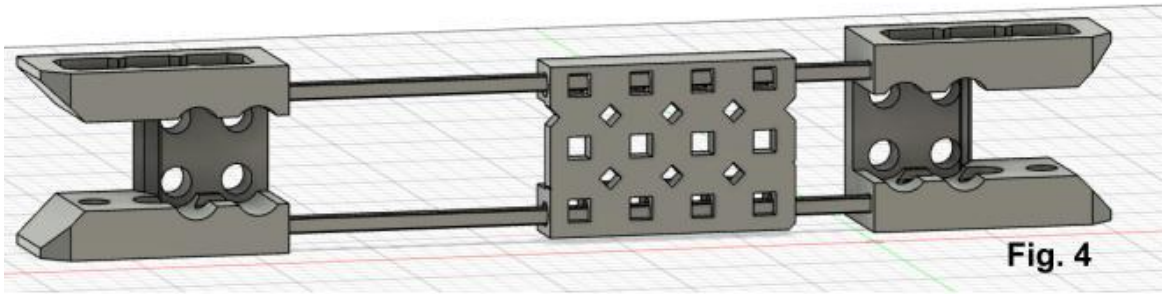
Fig. 4

Explanation of Creation of Perceptron Guide

(See Figures 5, 6, and 7)

For a detailed explanation of the creation of the Perceptron Guide, please see Appendix 1. We made the piece by first sketching a rectangle (using the point rectangle tool) and increasing the depth to make a rectangular box. After copying this box, we made a different size rectangular box. We then combined these three structures. Next, we imported a previously cut 1 x 3 steel bar and used the negative space to make the guides accept the motor. We created a sketch of the circle using the center diameter circle tool and then used the press pull tool to make a cylinder. We used this object to make the holes for the screws to pass through, then chamfer cuts to make changes to the design.

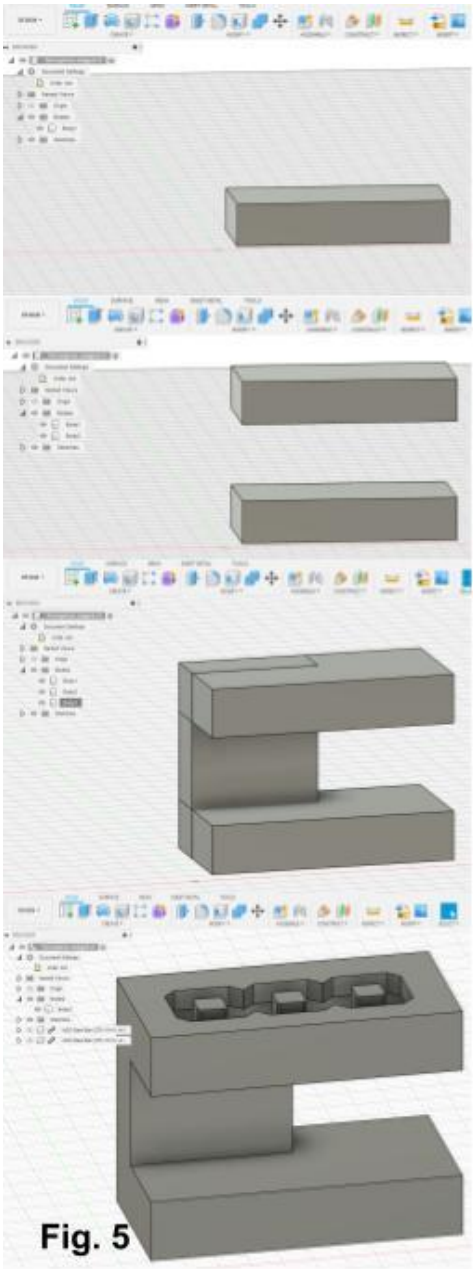

Fig. 5

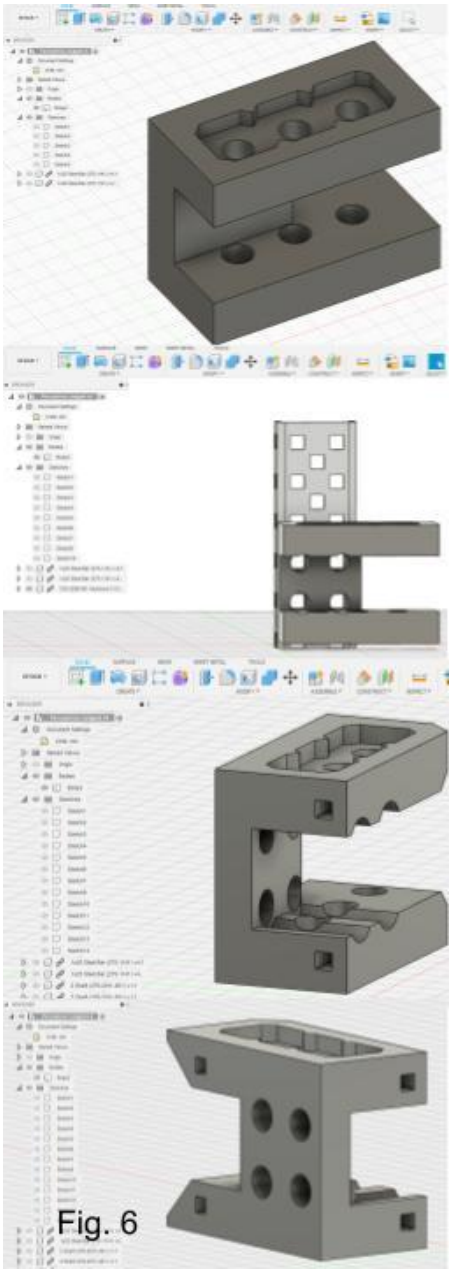

Fig. 6

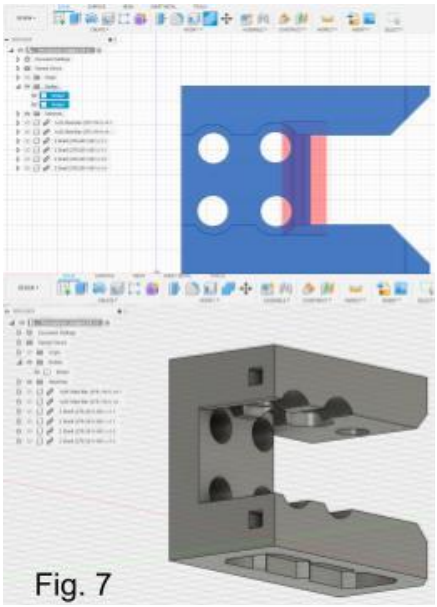

Fig. 7

How?

The first goal was to have SID perceive a shape (circle, triangle, or square) and create an array resembling the shape. The set up perception can be seen in Figures 8 and 9. The outcome of the reads can be seen in Figures 10, 11, and 12.

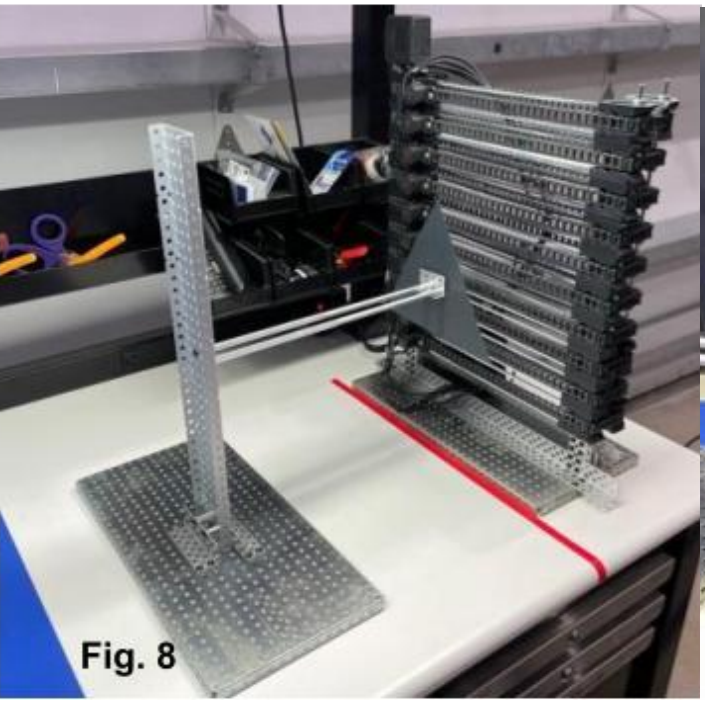

Fig. 8

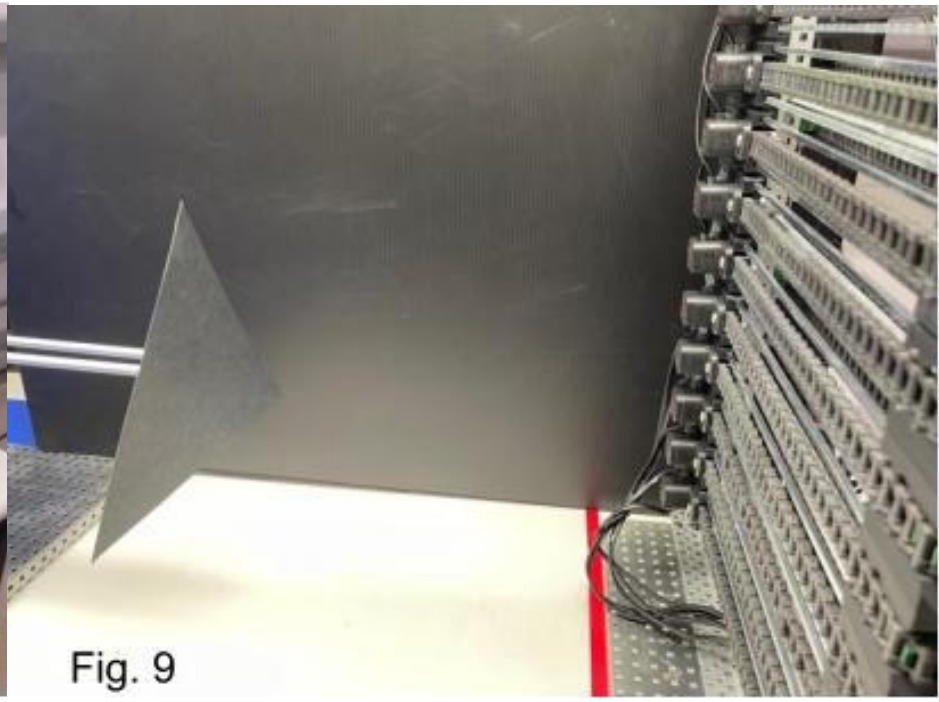

Fig. 9

```
747   Scanning row 18
748   Scanning row 19
749   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
750   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
751   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
752   0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
753   0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
754   0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
755   0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
756   0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
757   0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0
758   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
759
```

Fig. 10 Circle

```
312   Scanning row 18
313   Scanning row 19
314   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
315   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
316   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
317   0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
318   0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0
319   0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
320   0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
321   0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
322   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
323   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
324
```

Fig. 11 Triangle

```
988   Scanning row 19
989   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
990   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
991   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
992   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
993   0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
994   0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
995   0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
996   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
997   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
998   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999
```

Fig. 12 Square

Our second goal was to compare the previously read and labeled arrays to newly read ones. The two arrays in the code that are labeled are both the triangle shape; it had two slightly different arrays even though the triangle did not move. One of the sensors was likely at a tipping point and once measured a distance that was close enough, and once not. Using the code seen below (Figure 13), we were able to compare the arrays and show that the shape being perceived was in fact a triangle.

```
int firstTri[10][20] = {
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0},
  {0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0},
  {0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},
  {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
};
 int secondTri[10][20] = {
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0},
  {0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0},
  {0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0},
  {0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
};

for (int r =0;r<10;r++){
  for(int c = 0;c<20;c++){
    if(grid[r][c]!=firstTri[r][c]&&grid[r][c]!=secondTri[r][c]){
      std::cout<< "row" << r  << std::endl;
      std::cout<< "col" << c  << std::endl;
      std::cout << "different" << std::endl;
    }
  }
}
```

Fig. 13

Conclusion

We learned a lot from this project. In the past I had used Tinkercad, and for this project we switched to Autodesk Fusion 360. At first, I thought this change would be daunting, but it was the opposite. Fusion 360 is very intuitive. The amount of help from Autodesk directly, through forums created by Autodesk and from Youtube videos that the community of Fusion 360 users has created was impressive. The second area of the project we learned from was the programming of SID. Usually, programming a robot entails the use of sensors for movement. In this case the motors and sensors were creating a data array and then our goal was to analyze that array. Lastly, the creation process for the Perceptron Guide and shuttle for SID, made us think of other uses for this apparatus. Four Perceptron Guids could also be placed on the outside of a robot so that one motor could move numerous sensors to gain information about the surroundings. For example, a standard drive could have the Perceptron Guide find a line that is off to the side with the shuttle and then use the drive to center the line and follow the line. (Figures 14, 15, 16, and 17).

The lessons that this project taught will be of great value to us in the future. Using Autodesk Fusion 360 to model the piece in a way where we had to explain the build process taught us how to use the Fusion 360 efficiently. I foresee many on my robotics team and myself going into engineering, so becoming facile with Computer Aided Design will be very useful for our futures. Lastly, artificial intelligence is the wave of the future. By participation in the Autodesk Fusion 360 VEX Online challenge, we ventured into principles of Artificial Intelligence. Although we did not create a multilayered neural network that can solve for all prime numbers, our nascent foray has served us well.

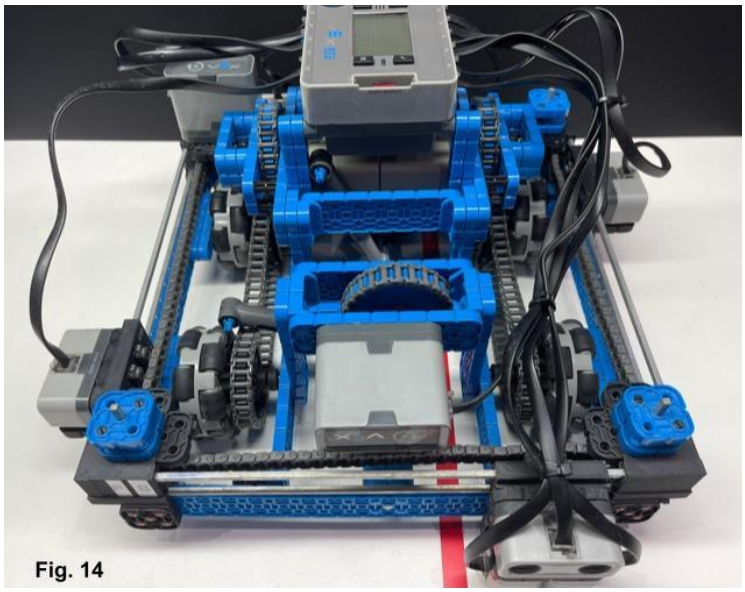

Fig. 14

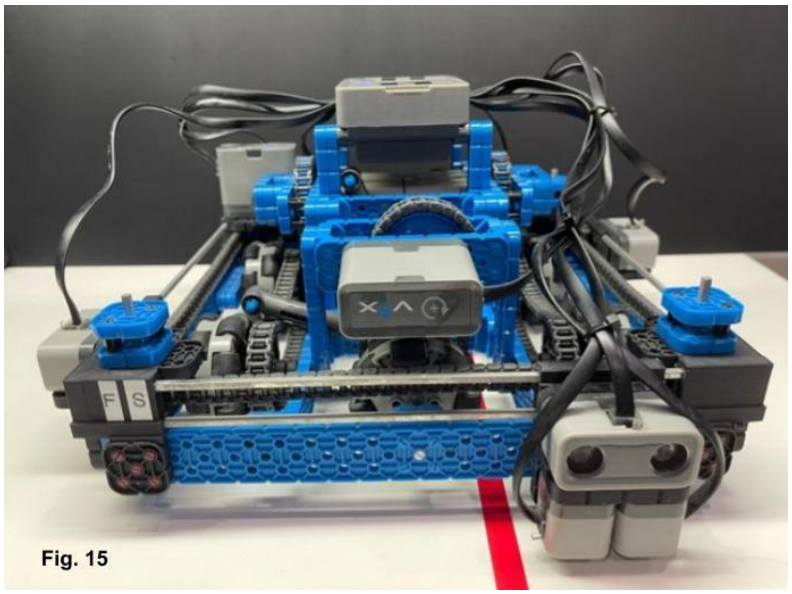

Fig. 15