Odometry

Instead of using encoder reading to determine the robot's current heading, we decided to use the IMU reading for robot heading. We soon discovered that the IMU returns 89 degrees for every 90 degrees it turns, which is around the +-0.01 degree per 1 degree error described on VEX's website. Thankfully, the IMU reading is consistent enough and we were able to add a scale factor to address this issue.

Our code for heading correction:

```
// the inertial sensor reads slightly less than 1 for each degree
double heading_correction (double currentRotation){
 double correctedHeading = fmod((currentRotation * scaleFactor_heading), 360) + init_heading;

 if (correctedHeading > 360){
   correctedHeading = fmod(correctedHeading, 360);
 }

 if (correctedHeading < 0){
   correctedHeading = 360 + correctedHeading;
 }

 return correctedHeading;
}
```

We defined our global coordinate system in the way shown below:

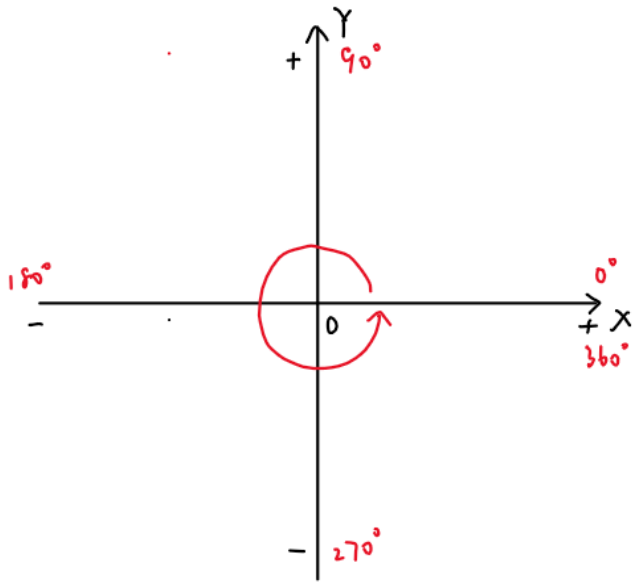


Figure 6: Coordinate system.

Each time the program goes through the loop, the robot moves in both X and Y direction. Therefore, the robot displacement can be decomposed into two vectors, each in the direction of the robot's *local* X and Y axis.

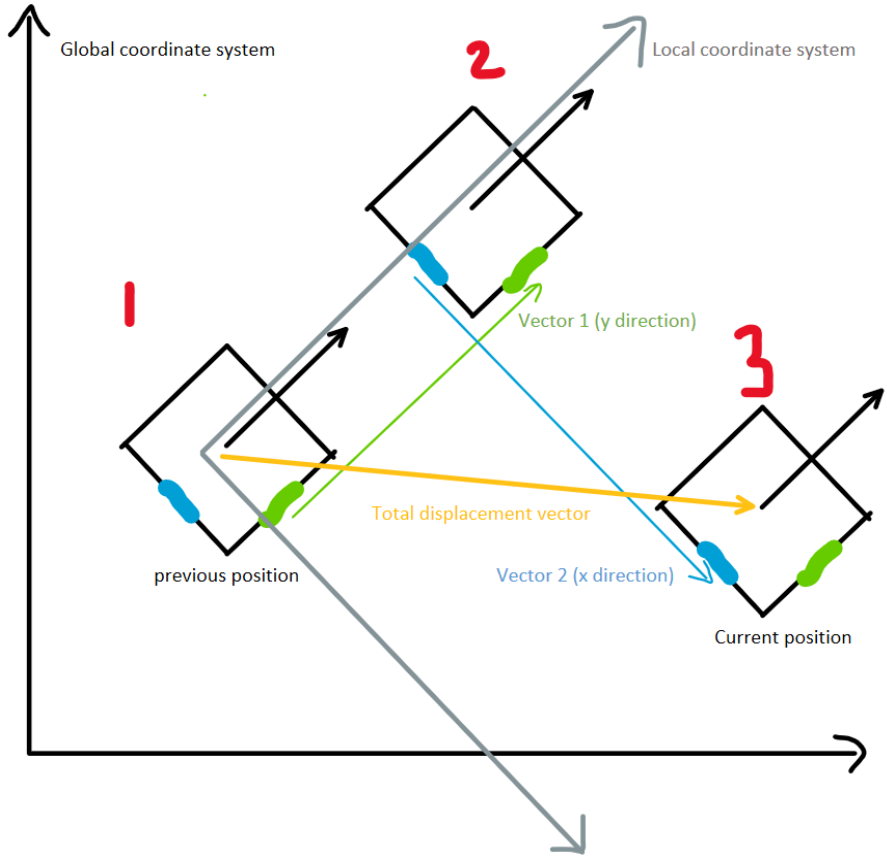


Figure 7:
local coordinate axis are shown in gray, global coordinate axis are shown in black

In the picture shown above, the robot starts at position 1 and ends up at position 3. This movement can be decomposed into two motions. That is, the robot first moves a certain distance in its local Y direction (the green arrow), then moves a certain distance in its local X direction (the blue arrow).

Since the robot's heading might change between each loop, it's more appropriate to model the green and blue arrows as arcs instead of lines, as shown in the figure on the right.

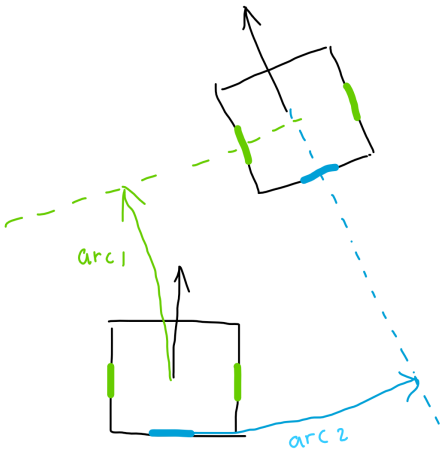


Figure 8: Illustration of robot displacement over a single time step.

We decided to break down the problem by looking at one direction at a time. At the end of each loop, the y direction path the tracking wheels on the robot traveled are shown in the picture below as red and green arcs. Since the tracking center is located at the center of the robot, it has traveled through the yellow arc, which has the length of (red arc + green arc) / 2. Therefore, the Y direction displacement of the tracking center is denoted by the yellow arrow. Now, it is reasonable to define a *local coordinate system* that has its y axis aligned with the yellow vector. This way, the displacement the robot has made in the global coordinate can be obtained by rotating this local coordinate system by a certain angle.

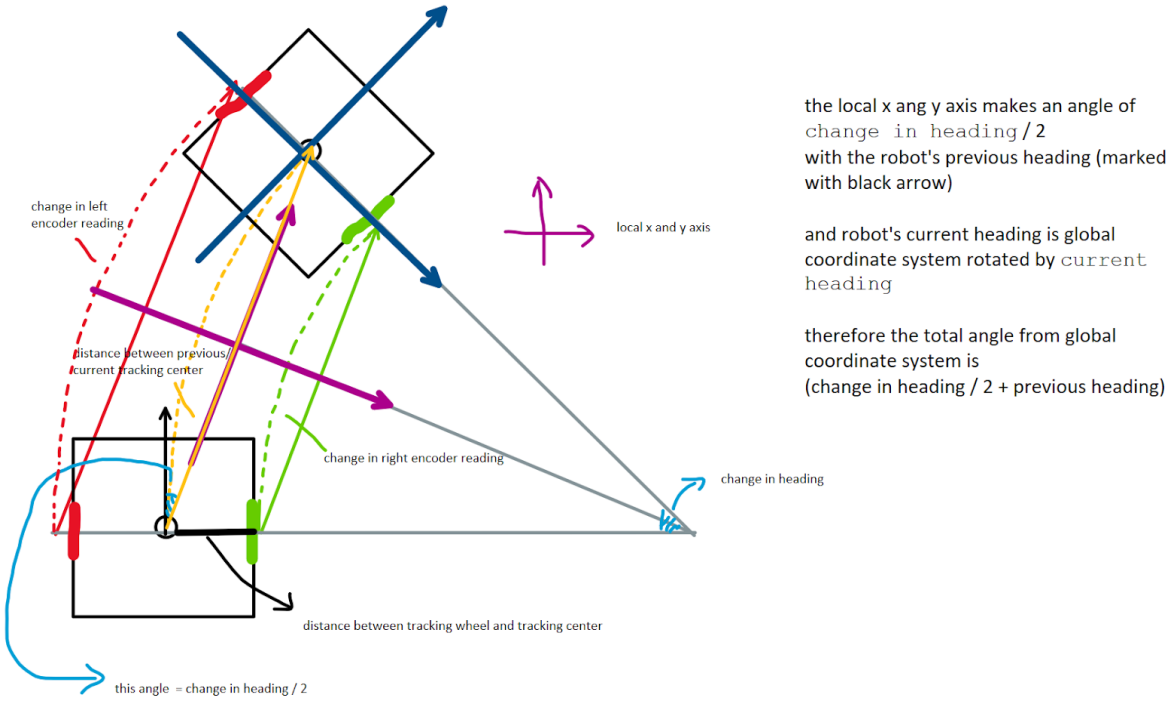


Figure 9: Change of coordinate system.

The length of the yellow vector can be calculated with some simple trigonometry.

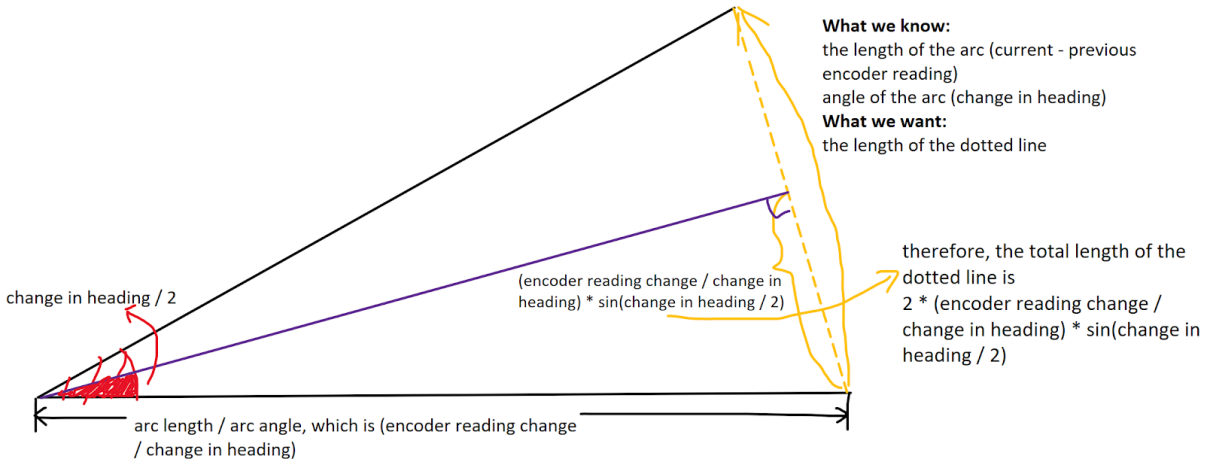
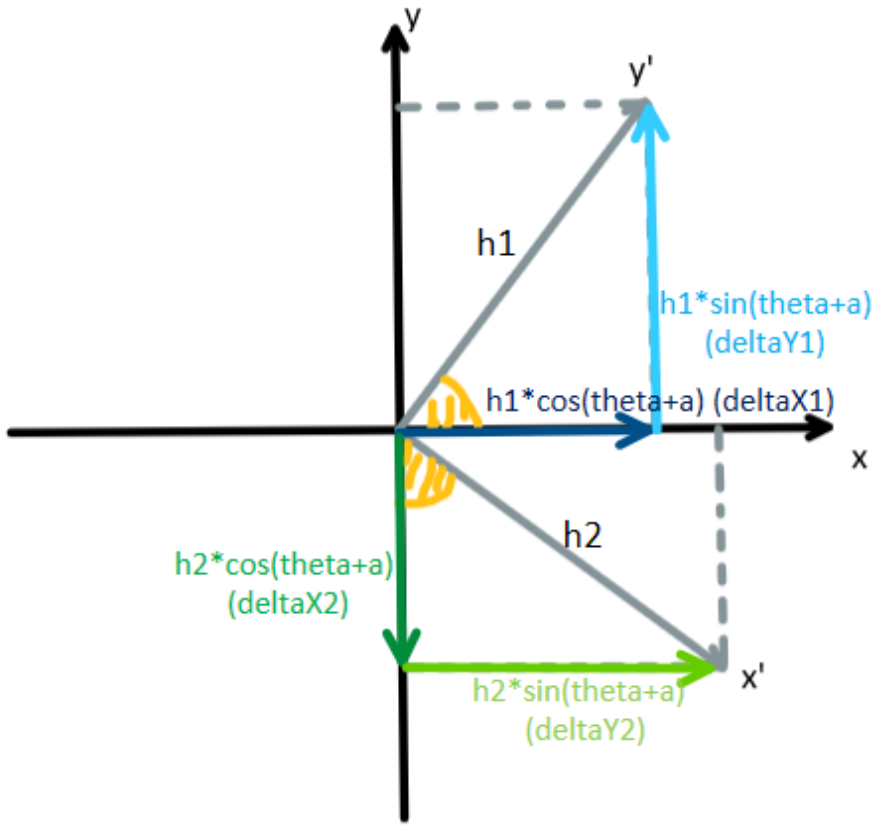


Figure 10: Computing x and y displacements from arcs.

The last step in this algorithm is to convert the displacement vectors from the local coordinate system to the global coordinate system. As previously calculated, the angle between the Y axis of the local coordinate system and the X axis of the global coordinate system is equal to (previous heading + change in heading / 2).



In the picture on the left, the gray axis represents the local coordinate system and the black axis represents the global coordinate system.

Therefore,

Global delta x = deltaX1 + deltaY2

= h1*cos(theta+a) + h2*sin(theta+a)

Global delta y = deltaY1 - deltaX2

= h1*sin(theta+a) - h2*cos(theta+a)

Where h1 is local y displacement and h2 is local x displacement.