RoboCup Junior Australia

Introduction to Omni Drive Systems

For RoboCup Soccer Competitions

William Plummer

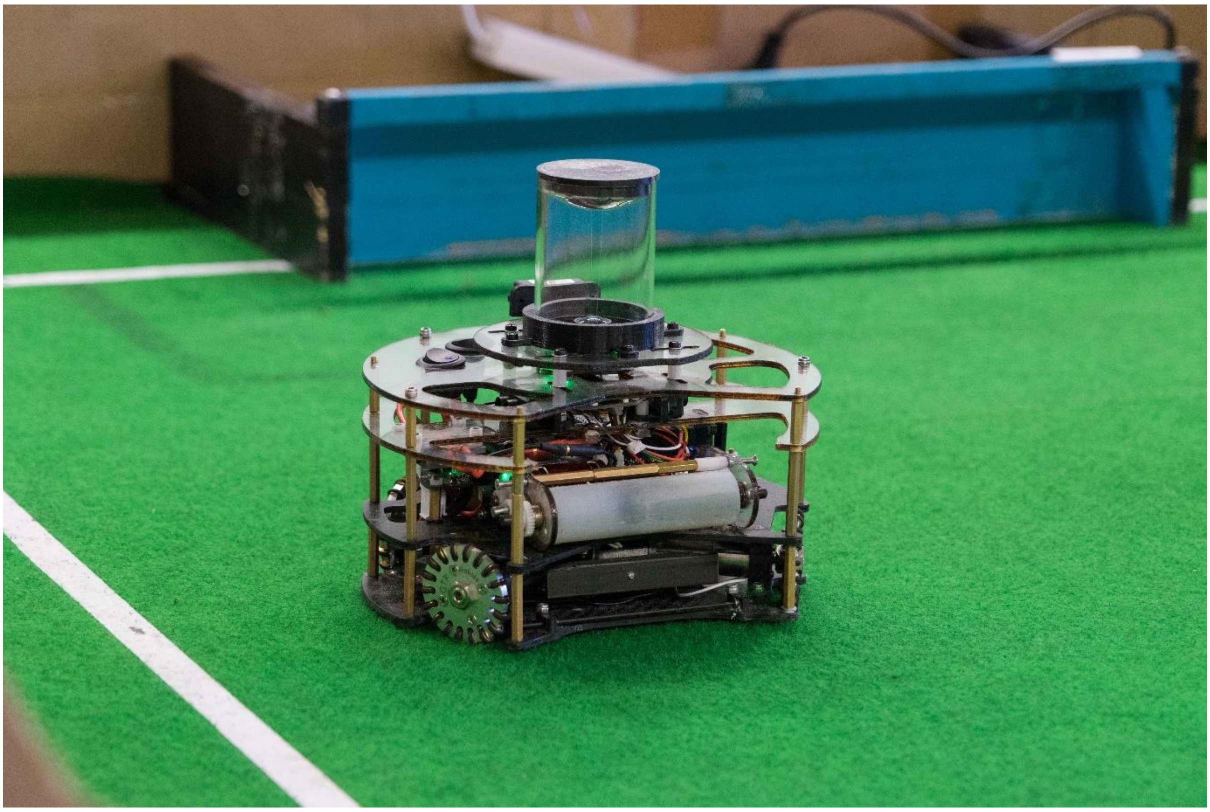


Table of Contents

Overview

You may have seen the teams in the RoboCup Soccer competition using omni wheels, such as the ones pictured below:

What are the reasons for teams using these?

- Omni wheel robots allow for movement in all directions at any given time, without having to rotate to execute that movement
- With a correctly setup omni drive system, a direction and speed can be input and translated to motor speeds for all motors simultaneously, for any number of motors at any angle
- Allows for smooth ‘orbiting’ around a soccer ball while playing – this sweeping motion allows the robot to quickly get to the ball in play

A proper omni drive system can be very simple or complex, depending upon how many variables you want your robot to factor into its algorithm. This document will start with a simple, theoretical system then adding modules and sensor inputs such that it is more effective in realistic situations, like a RoboCup Soccer competition.

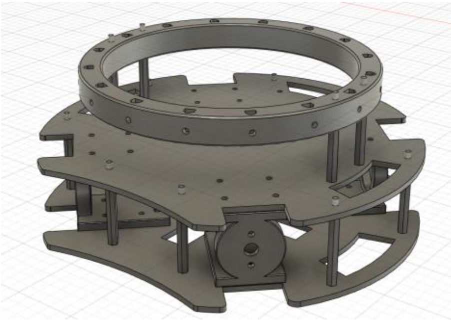
Modern Teaching Aids has Omni Wheels available for purchase through their website:

[Dual 50mm Rotacaster – 35A – LEGO Hub – Each - MTA Catalogue (teaching.com.au)]

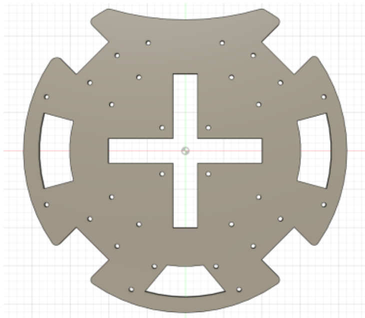
[Omniwheel Holonomic [Non LEGO] – EACH - MTA Catalogue (teaching.com.au)]

Simplified Model

Omni drive robots are generally designed around circles, as this allows for an easily identified centre point and helps with weight distribution over each of the motors. Furthermore, we assume or design the robot such that the motors and wheels are all located the same distance from the centre point. This is not nessicary for omni drive to work, however it significantly simplifies the maths involved. The same is true when using a circular design, other shapes will work, but will require more maths. For the example, the robot will use 4 motors positioned at 45 degrees offset from the front of the robot, shown below:



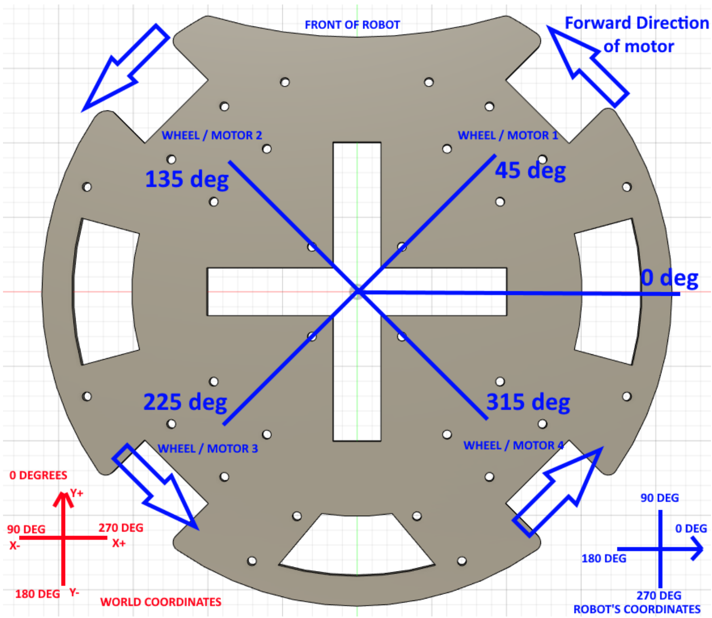
Orthographic View



Top View of Bottom Plate

This base plate has multiple functions, the three radial cut-outs within the plate are purely for weight saving, the cross in the middle is to house a light sensor array and the holes all over are for mounting standoffs. This leaves the four cut-outs along the edge of the plate – these are spaces for the omni wheels, with the motors to be mounted to the four holes directly behind. Note how the wheel cut-outs are 45 degrees offset from the front capture zone centre point and are 90 degrees apart from each other.

First, we will make theoretical model of the robot. To initialise this, we must make two coordinate systems, one for the world around the robot (the soccer field for example) which will use RED letters in the diagram. The second coordinate system is the one relative to the robot, which will use BLUE letters.



Now onto the maths. We will start with the assumption that the top of the image is the front of the base plate and that it is facing 0 degrees in the world coordinate system. We will assume that for the robot's coordinate system that directly right is 0 degrees, increasing anticlockwise. We will also wire the motors such that setting their direction to forwards, they will spin anticlockwise. This is also very important.

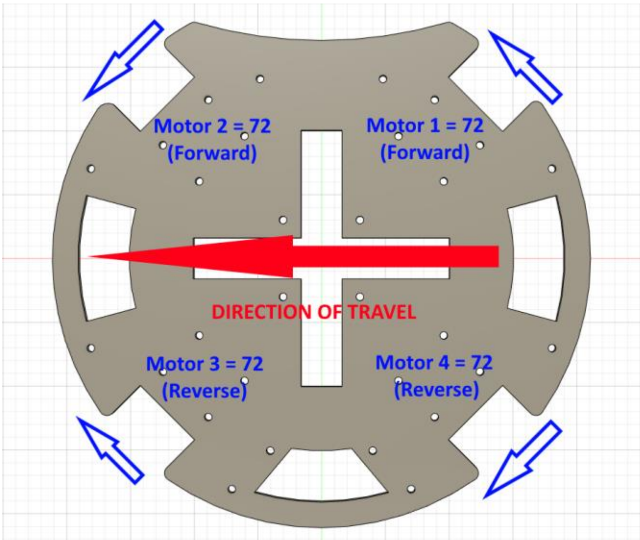
Next, determine the components of movement each motor will provide depending upon their speed. There are two ways of doing this, either with sin or cosine, however cosine has been used here to help differentiate between the two coordinate systems as sin has the robot's 0 degrees at the same position as 0 degrees on the world coordinates. Also note, we are using standard PWM motor controllers for this example, which require a 0-255 speed and a separate value for direction (01 or 10). **These first equations only generate the speed, not the direction.**

$$Motor\ x = abs\big(cosine(motor\ angle - \ desired\ direction)\big) \times 255 \times desired\ speed$$

For the example bottom plate travelling at 40% max speed with a desired direction of 90 degrees (world coordinates), the equations would be:

$$\begin{aligned}
Motor\ 1 &= abs(\cos(45 - 90)) \times 255 \times 0.40 \\
Motor\ 2 &= abs(\cos(135 - 90)) \times 255 \times 0.40 \\
Motor\ 3 &= abs(\cos(225 - 90)) \times 255 \times 0.40 \\
Motor\ 4 &= abs(\cos(315 - 90)) \times 255 \times 0.40
\end{aligned}$$

Which results in these motor speed and directions, which drives the robot to the left:



For determining the direction that the motor should spin, simply use a logic check to whether the value of $\cos(\text{motor angle} - \ \text{desired direction})$ is positive or negative. Also ensure that the value that you output to the motor controller is a whole number (0-255).

These equations will work for any circular vehicle with any number of motors, at any given angles, such that they are equally distanced from a single centre point. However, different numbers and angles of motors will result in different movement characteristics. For example, if the motors are closer to the sides of the robot, the forward and reverse speeds will be improved but at the cost of a loss in horizontal movement.

Model Refinement

Although the movements these equations generate are correct, they assume that the motors are perfectly equal, that the ground has the same amount of friction in all areas and from all directions and that the robot will never experience any other outside forces. We can address these real issues with additional calculations in the model.

Speed Floor

When the robot attempts to move with its motors, it must first overcome the stall friction that the ground exerts upon the wheels. This is essentially a baseline load that is factored in into the equation and is different for every robot and can be found as the maximum speed the robot's motors can be running at while stationary with no external force.

For the theoretical robot, we will use a stall speed of 15% for example, meaning that at 15% of the max PWM 255 (38), the robot will be stationary but will start moving at 39 PWM speed. However, it is not as simple as adding the baseline speed, as we must scale down the omni drive speed to account for the speed floor. For example, with a stall speed of 15% and some direction the robot wanted to travel required 90% of the maximum motor speed, the robot would need to run that motor at 105%, which isn't possible. We scale the motor speeds as such:

$$Motor\ x = 255 \times (Speed\ Floor + (1 - \ Speed\ Floor)(abs(\cos(motor\ angle - desired\ direction)) \times desired\ speed))$$

Which would look like this with the example so far:

$$\begin{aligned}
Motor\ 1 &= 255 \times (0.15 + (1 - 0.15)(abs(\cos(45 - 90)) \times 0.4)) \\
Motor\ 2 &= 255 \times (0.15 + (1 - 0.15)(abs(\cos(135 - 90)) \times 0.4)) \\
Motor\ 3 &= 255 \times (0.15 + (1 - 0.15)(abs(\cos(225 - 90)) \times 0.4)) \\
Motor\ 4 &= 255 \times (0.15 + (1 - 0.15)(abs(\cos(315 - 90)) \times 0.4))
\end{aligned}$$

Compass Correction

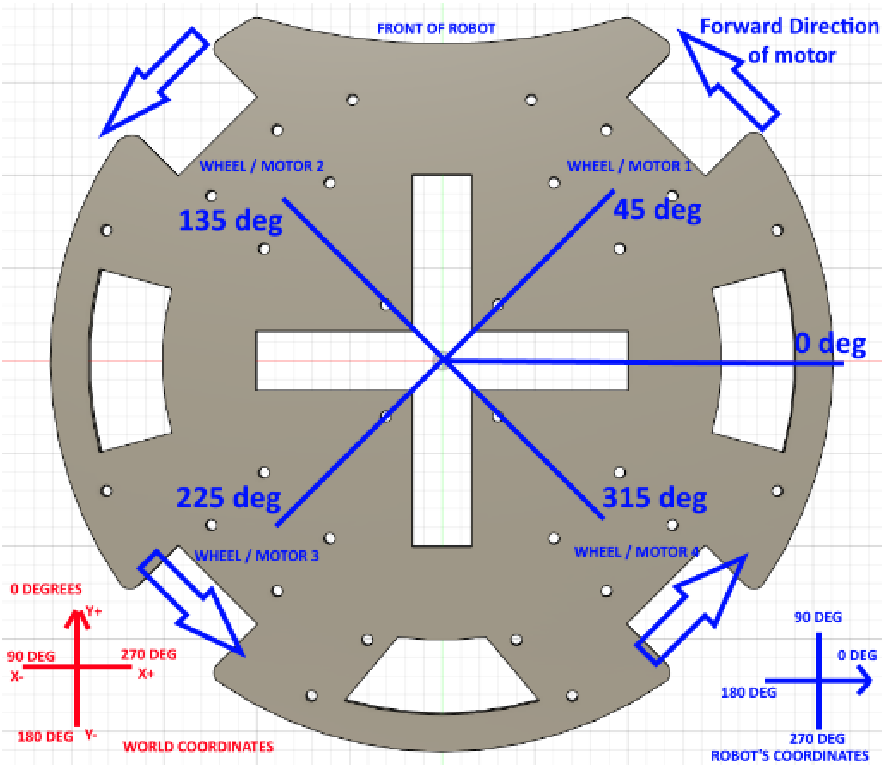
Another major problem to correct for is if the robot is rotated unexpectedly, requiring correction. There are multiple different sensors that can give a bearing such as:

- Compass Sensor
- Inertial Measurement Unit (IMU)
- Gyroscope
- Magnetometer

Note that these sensors have their strengths and weaknesses, research into them to determine which would be best for your application.

By taking a reading for a local 'North' with one of these sensors, the code can be significantly simplified by setting that local 'North' heading as 0 degrees on the global coordinate system. Further, that local 'North' should be the same direction as your goal to your opponent's goal.

Remember that the motors are all forwards in the anticlockwise direction:



By spinning all motors forwards, the robot will rotate on the spot anticlockwise.

By spinning all the motors backwards, the robot will rotate on the spot clockwise.

Using this, we can simply add on a speed to all the motors to get the robot to rotate while in motion, using the same method of scaling as the Speed Floor, show below:

$$\begin{aligned} Motor\ x = 255 \times (&Speed\ Floor + (Compass\ Correction\ Value * Compass\ Correction\ Strength) + \cdots \\ &(1 - Speed\ Floor\ Compass\ Correction\ Strength) \times \ldots \\ &(abs(\cos(motor\ angle - desired\ direction)) \times desired\ speed)) \end{aligned}$$

As for determining what the compass correction value is, this can at a basic level be done by using the difference between the initial 'North' reading and the current reading. The bigger the difference between the current heading and the desired heading, the stronger the correction is. Note that a complex polynomial equation to better model a response will give better results, but requires a lot more time, maths and is different for every robot.