BRIEF DESCRIPTION OF CONTROL SYSTEMS OF EKLAVYA 4.0

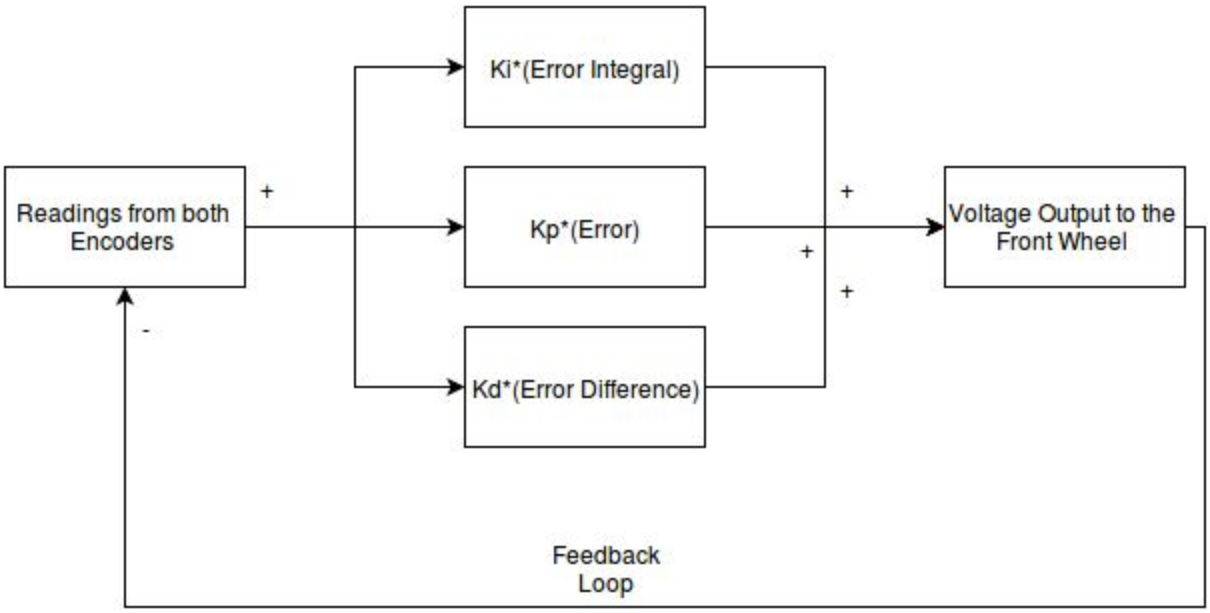
The entire controls modules run on the laptop connected to the bot and the hardware responsibilities are met by a pair of Arduinos. There is a closed loop PID mechanism that is used to control the hub and the steer motors.

There are 2 simultaneous PID (Proportional, Integral and Differential) loops in the code. One of them controls the Vx (The velocity in the direction perpendicular to the axis) and another controls the instantaneous curvature C of the robot. Curvature of the robot is given by the inverse of the radius of curvature.

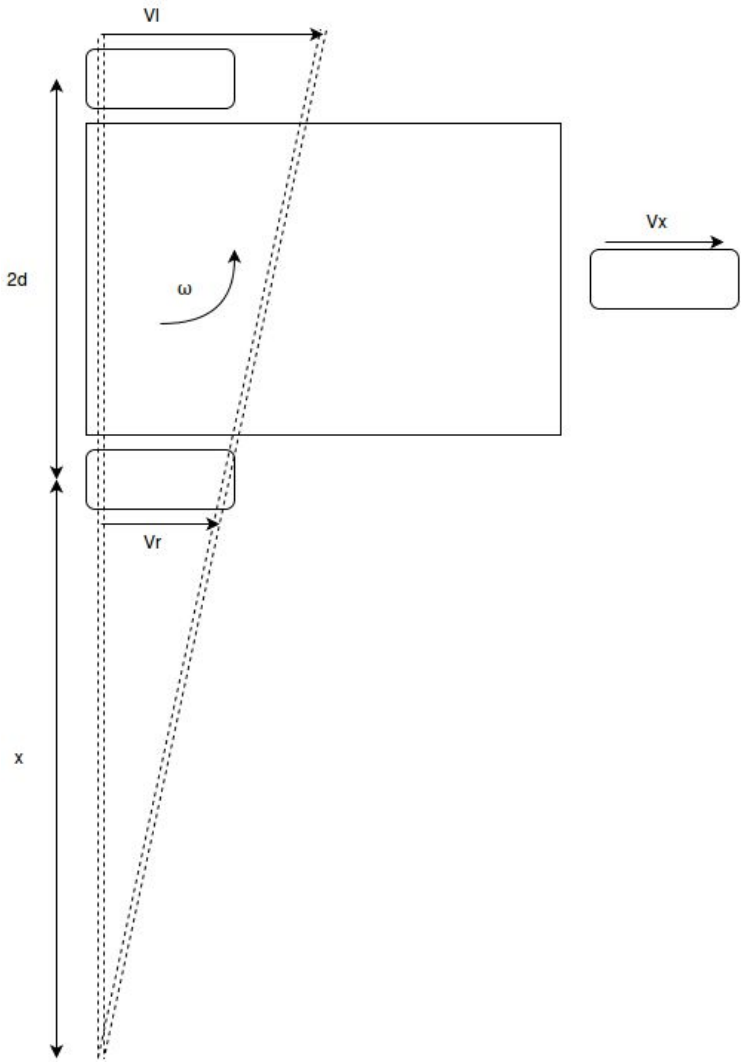
The sensor readings for the Vx are given by the encoder that are attached to the axle of the wheels. The average of the two readings from the left and right encoders gives a measure of the velocity in the x direction which is used as the independent variable in the loop, while the derived variable is the value of the analog voltage that is given to the hub motor, which in turn is directly proportional to the target velocity that needs to be achieved.

The sensor readings for curvature are derived using the angular velocity readings from Vectornav IMU unit. The IMU sends its data on a Robot Operating System (ROS) topic, which the control node on the same system subscribes to. The IMU sends the angular velocity of the robot about its own centre axis, perpendicular to the ground and the axle.

The mechanics that govern the motion are assumed to be pretty simple. We consider the robot makes perfect circular turns with varying radii of curvatures. We assume negligible slipping in the wheels and assume that the encoder readings are pretty accurate all the time.



Using these certain approximations, the equations for motion are as follows



Vl=Left wheel velocity
Vr=Right wheel velocity

ω=Angular velocity

d= half the distance between two wheels

R= radius of curvature

x= distance b/w the right wheel and center of curvature

Equations:

$$v_x = \frac{(v_l + v_r)}{2} \tag{1}$$

$$v_l = v_x + \omega.d \tag{2}$$

$$v_r = v_x - \omega.d \tag{3}$$

using equations 2 and 3, we get

$$v_l - v_r = 2\omega.d \tag{4}$$

and we also know that

$$\frac{v_r}{x} = \frac{v_l}{2d + x} \tag{5}$$

which implies,

$$x = 2d\frac{v_r}{(v_l - vr)} \tag{6}$$

$$d + x = d\frac{(v_l + v_r)}{(v_l - v_r)} \tag{7}$$

replacing $d + x$ by r and using 1 and 4

$$r = \frac{v_x}{\omega} \tag{8}$$

Using these equations, we apply two types of PID equations.
In the first equation, the derived result is the voltage to be given to the motor and the feedback value is the Vx, i.e. the average of the two readings of the encoder. The data is given out in the form of a 12 bit DAC number which in turn is used by the Arduino Due to convert into a voltage value. We also used a bias of 1400 which is the value of the DAC at which the motor starts.

The second PID is to give the angle of rotation of the front steering column. The derived variable is the angle to be given to the Roboteq controller, which in turn uses

another internal feedback loop to ensure that the motor turns to the desired value. The alpha is in the range -1000 to 1000 for values from -45 to 45. The feedback variable was chosen to be the curvature in this case. It was so because Radius of curvature tends to infinity when the bot is moving straight, while curvature doesn't tend to infinity for any relevant part of the motion.

Using these two equations, we tuned the PID. The corresponding values of PID constants are:

DAC_value=410*Vx_error+6*Vx_Error_integral+110*Vx_error_diff+1400

Alpha_value=380*C_error+20*C_Error_integral+500*C_error_diff

These both values are sent via ros topics to Arduino Due and Roboteq respectively, which in turn drives the robot.