Type of Drive

According to the task of the competition we have decided to build a mobile platform which should be able to move in all directions and rotate independently. MR1 has to travel through the forest according to the route indicated with white guidelines, for which we also need to optimize the robot's reaction speed and thus decided to build our robotic base equipped with holonomic drive rather than differential drive. Selection of the number of wheels was our first decision. Two wheel drive has simple design and control but reduced manoeuvrability. 3WD is a little bit complex but has good traction. 4WD has more complex mechanics and control. Finally, we come to the conclusion that a three-wheel platform should be a solution to follow due to its following advantages:

- It offers greater traction as any reactive force is distributed only through three points and the base is balanced even on uneven terrain.
- This design structure is less complex and offers rich manoeuvrability.
- This design also reduces additional wheels and other components and make cost-effective.

Kinematics of 3WD mobile platform:

The robot should be able to move in a particular direction given as an input by the joystick. The joystick gives values in both directions (horizontal and vertical). Thus I get a vector as input. Let us say the vector along the direction is $\vec{D} = x\hat{\imath} + y\hat{\jmath}$ where i and j are the unit vectors along the directions horizontal and vertical respectively. The magnitude of the vector would be $|\vec{D}| = \sqrt{(x^2 + y^2)}$. Now the platform should move in the direction $x\hat{\imath} + y\hat{\jmath}$ without rotating. Consider the forces acting on the body corresponding to each wheel due to the torque generated by its motor are $\vec{A}$, $\vec{B}$ and $\vec{C}$ respectively and the direction of these vectors are perpendicular to the axis of the wheel and parallel to surface. Now equate the force vectors of each wheel with the input vector $\vec{D}$. Also the platform should not rotate, so we need to equate the torque of the system to zero. From these two equations, we get the ratio of forces acting on each wheel in terms of the inputs (x and y). Considering that the torque of the motor is proportional to the input PWM signals, we get the ratio of the duty cycle of input PWM signals in terms of inputs x and y.

The required relation is:

$$\overrightarrow{|A|} : |\overrightarrow{B}| : |\overrightarrow{C}| = 2\,\frac{x}{3} \;:\; \frac{x}{3} - \frac{y}{\sqrt{3}} \;:\; \frac{x}{3} + \frac{y}{\sqrt{3}}$$

And the speed of the platform should be proportional to the magnitude of the input vector $|\vec{D}|$. In the code we can simply map our PWM values corresponding to the magnitude and this equation will automatically calculate the ratio of PWM values for that particular direction input.

Implementation of PID for Motor Control:

Now I have successfully calculated the required velocity of each wheel for the desired direction but in this type of configuration the whole system will be disturbed if any wheel is not rotating with the calculated RPM and this would always happen as practically various parameters get affected(battery voltage, the resistance of the coil, etc). So to compensate for this problem we need to control the RPM of a motor by taking some feedback and we decided to implement PID on motor control.

For taking input we have used an optical rotary encoder to measure real-time RPM of the motor shaft. These signals are given to the microcontroller and actual RPM is measured. The set point (input RPM) and the actual RPM are compared to get error e. From this error, the PWM is calculated by the PID formula.

PID algorithm:

1. Calculate the error (error = reference speed – input speed).
2. error_sum = error + error_sum.
3. Calculate PWM by the formula.
4. Previous_error = error.
5. Repeat.

PID formula:

$$PWM = Kp * error + Kd * \left(error - error_{previous}\right) + Ki * (error + error_{sum})$$

This signal is directly given to motor driver and the error is compensated. The constants in our case are:
Kp = 0.00545 ; Kd = 0.01 ; Ki = 0.0099 .

These values are calculated from Simulink PID tuner by providing some parameters which are available on a data sheet of a motor.

Thus the shaft will always rotate at constant RPM even if the load is varied.

Implementation:

I have used ISR of a microcontroller for reading quadrature optical rotary encoders. The resolution of this encoder is very high (2000 CPR). Due to such hugh resolution, that signals from other encoders attached to lower priority may get missed when the ISR is being executed. To resolve this issue I have use a separate microcontroller (Arduino Nano) to control each motor. The work of these microcontrollers would be to control the RPM of a motor with PID. There would be the main microcontroller which would calculate the PWM values of each motor for the given input direction and these values would be sent to the all separate microcontroller through i2c communication.

Controlling:

We could not use a wired controller as wire may get stuck at the forest zone during the game play. So I decided to use a wireless controller. We have used the X-Box 360 controller. It is comfortable to use and has smother analog controls. We will be using one joystick for translational motion of the base in any direction and other for rotational motion. The control bus uses 10-bit commands and can be easily read by our microcontroller.