2020-2021
VISTA RIDGE HIGH SCHOOL
VICTORIAN VOLTAGE
CONTROL AWARD

FTC 7797

Control Award, Sponsored by Arm, Inc., Submission Form

****Please turn in this sheet during your judge interview along with your engineering portfolio****

Team #7797	**Team Name:**Victorian Voltage

Required for Remote Submissions – Please provide a link to a video recording of the controls described in this submission form:
https://youtu.be/3iGmaNigyRs

Autonomous objectives:
Our goals in autonomous is to bring both wobble goals to the respective zone based on the ring stack. And to shoot the 3 rings from our robot into the goal.

Sensors used:
Distance sensors: two in the front one in the intake.
Deadwheel odometry

Key algorithms:
Kalman Filters, PID, Motion Profiling, Saftey checks.

Driver controlled enhancements:
Autoswitching our intake system from lower intake to shooting.
Precision driving and auto claw grabbing.

Engineering portfolio references:
The software section of our portfolio.

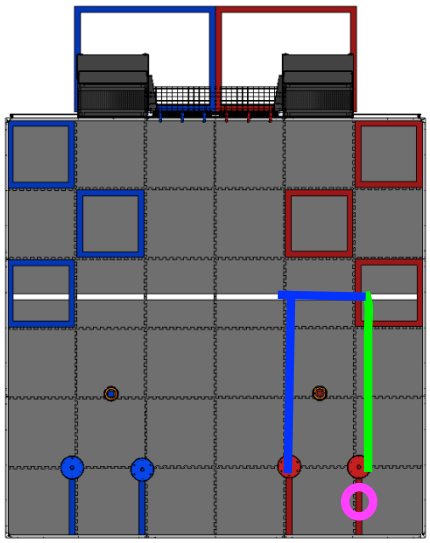
Autonomous program diagrams:

The pink circle is where we sit and scan the amount of rings.

The green line is the path that the robot will follow to drop of the first wobble goal

The blue line is the path the we follow to get our second wobble and return it

We then move to the middle of the goal and go behind the line to shoot of our rings.
This pathing is a generalization for all 3 rings variations.

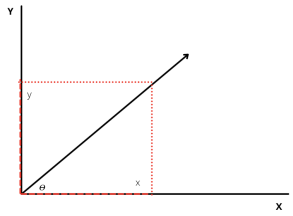


- Positioning

  A vital part of having a successful autonomous is having our robot know where it is on the field. To help tackle this problem, we used Dead Wheel Odometry, a Kalman Filter, and robot localization.

  - Dead Wheel Odometry

    The encoders on the motors are limited in their capabilities. This is because encoders are attached directly to the motors; they are not directly measuring robot movement which can cause issues with detecting real change in position. To counteract this we use a system called “Dead Wheel Odometry” to tackle this. A *Dead Wheel* is an unpowered wheel that spins freely as the robot moves. This wheel is attached to an encoder (Vex SRX Mag Encoder), which allows us to get the positional matrix X = [x, y, θ]. Where “x” is our lateral position, “y” is our forward position, and “θ” is the angle for the robot coordinate plane, shown in the picture below.

  - Kalman Filter

    In addition to our Dead Wheel localization system, we use a Kalman Filter. Also known as a linear quadratic estimator, a common control system technique used in industry-standard robotics. The Kalman Filter breaks down into two parts; a prediction and an update.

    The prediction step estimates how far the robot has moved within a time interval. We use our predictions state and covariance plus the data from our dead wheels to estimate how far our robot has traveled. With each state, we use the last estimate guess to update our predictions.

- Movement

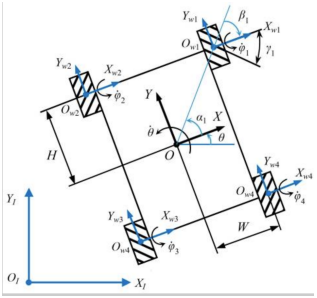
  Understanding how our robot moves will allow us to have a better understanding of where our robot is on the field and where it's going to move to in the future. This more accurate state of our robot can be plugged back into our positional control system giving us a more finely tuned model.

  - Kinematics

    Due to the nature of this year's game we decided to use Mecanum wheels to allow our robot to move in any direction. To allow our robot to move in any direction our Mecanum wheels which require wheels to rotate at different rates and directions depending on direction of robot movement. The following kinematics equation translates desired speed and direction in the robot coordinate plane to angular velocity of wheels.

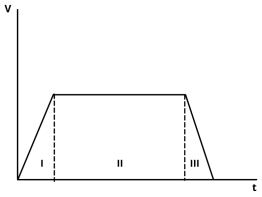
$$\begin{bmatrix} \dot{\varphi_1} \\ \dot{\varphi_2} \\ \dot{\varphi_3} \\ \dot{\varphi_4} \end{bmatrix} = \begin{bmatrix} -1 & 1 & W+H \\ 1 & 1 & -(W+H) \\ -1 & 1 & -(W+H) \\ 1 & 1 & W+H \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

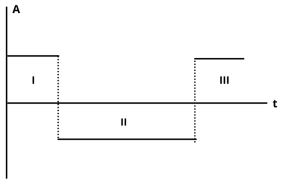
    Where φ_n is the power applied in a specific direction, ‘W’ is the width of the robot from the center to the middle of the wheel. And the ‘H’ is the height of the robot from the center to the middle axle. The rate of rotation of the robot determines theta dot and the x dot and y dot base on our error unit vector. To make sure that the output power is scaled to be within the range of 0 to 1. We divide by absolute maximum value.



(From Kinematic Modeling of a Combined System of Multiple Mecanum-Wheeled Robots with Velocity Compensation)

- Motion Profiling
  To help reduce slipping caused by abort changes in movement, we decided to use a technique called Motion Profiling. Which works by zoning our robot's movement into zones and fluctuating the power based on which zone the robot is in. If our robot is in zone one (ruffly the first quarter of the distance), we know that our robot will accelerate. In zone two, we know that our robot will be going at a steady velocity, and when we hit zone 3 (the last quarter), we know that our robot will start to decelerate.





Velocity v Time Graph of our robot. Acceleration V time graph of our robot

- PID control
  To help limit the uncontrolled error caused by our robot sliding due to inconsistency on the mat, we use a PID (Proportional Integral Derivative) control loop, which works by taking the error that we get from the robot and plugging it into this formula below.

$$Correction = k_P\, E + k_I \int E dt + k_D \frac{dE}{dt}$$

  Where k_p, k_i, k_d are all proportional constants and E is our error. Our constant values are discovered through rigorous testing of our robot. Then the correction is plugged into various other control system inputs that affect how our robot moves.

- Safetynet Exceptions
  Having a way to stop our robot in the case of an error is an easy way to limit the risk of damages caused to our robot.
  - Heartbeat
    The heartbeat exception is a simple exception that checks if there is still a "pulse" in our autonomous. During which our robot checks if our selected operation mode is actively running. In the case of our robot discounting due to static, we found that our robot would still attempt to drive forward even after acknowledging that the opmode was disabled.

  - Watchdog
    Our Watchdog exception gives oversight to check how long an activity is running if we notice that it is going over that designated time. The Watchdog will throw an error that will stop the robot and end the autonomous in its track.