Fall 2023

Mid Term Project Phase 0

Software Development for Robotics

October 17, 2023

Students:
Sameer Arjun S (119385876)

Manav B Nagda (119133545)

Ishaan S Parikh (119135891)

Instructors:
Tommy Chang

Course code:
ENPM808X

Contents

1 Overview **3**

2 Design and Development Process **3**

3 Ackermann Model Geometry **4**
3.1 Formulas 4
3.2 Usage 5

4 Process Model **5**
4.1 Class Diagram 5
4.2 Activity Diagram 6

5 Algorithm and methodology **6**

6 Tools and resources **7**

7 Assumptions **7**

8 Potential risks and mitigation **7**

9 Final deliverables **7**

List of Figures

1 Quad chart 3
2 Class diagram 5
3 Activity diagram for PID controller 6

1 Overview

The ACME robot company is creating a mobile delivery robot for a warehouse operation. This robot code named Valkyre has 4 wheels and is steered using an Ackermann steering mechanism. Valkyre is expected to enter service in major warehouse environments early next year. Our team is tasked to design the steering control software module to be added to Valkyre's control software stack

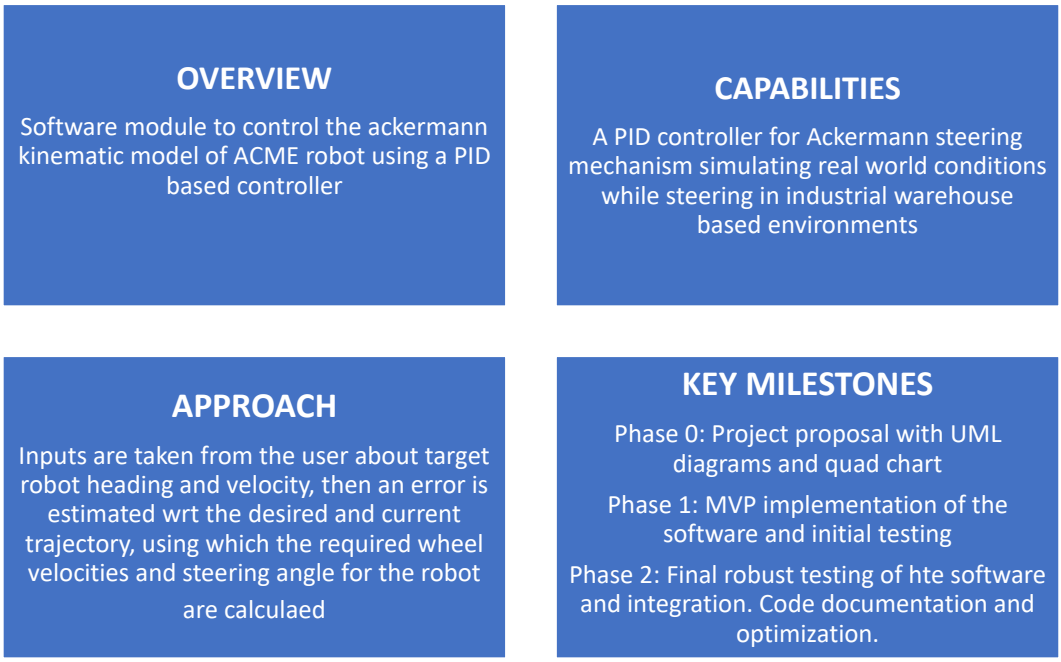

Figure 1: Quad chart

2 Design and Development Process

The team has identified the control mechanism for achieveing the Ackermann kinematic model with a max steering constraint of 45 degrees as provided by ACME. The module gets the input of the robots target heading and velocity to output the steering angle and drive wheel velocities to achieve the required steering motion for the robot. The classes of the future program and their responsibilities have been identified and their relations have been explained through UML class diagrams below.

The software team will work on this project through iterative software evolution and service processes and Agile Iterative Process (AIP) of software development will be used in the development process with Test-Driven Development approach. The team consists of 3 members and pair programming is implemented with one member working as the design keeper.

3 Ackermann Model Geometry

The Ackermann Model Geometry provides formulas for understanding vehicle dynamics when turning around a point with a given steering angle (θ) and time (t). These formulas are useful for modeling and analyzing vehicle behavior during turns.

3.1 Formulas

1. Turning radius of the vehicle:

$$R = vt$$

- The turning radius (R) is calculated as the product of the velocity (v) and the time (t) taken for the turn.

2. Inner wheel steering angle:

$$\delta_i = \arctan\left(\frac{L}{R - \frac{T}{2}}\right)$$

- The inner wheel steering angle (δ_i) is calculated using the arctangent function, where L is the wheelbase, R is the turning radius, and T is the track width.

3. Outer wheel steering angle:

$$\delta_o = \arctan\left(\frac{L}{R + \frac{T}{2}}\right)$$

- The outer wheel steering angle (δ_o) is calculated using the arctangent function, where L is the wheelbase, R is the turning radius, and T is the track width.

4. Distance traveled by the car along the arc for the inner wheel:

$$S_i = (R - \frac{T}{2}) \cdot \theta$$

- The distance traveled by the inner wheel (S_i) along the curved path is calculated as the product of the turning radius(R), minus half of the track width ($\frac{T}{2}$), and the heading angle (θ).

5. Distance traveled by the car along the arc for the outer wheel:

$$S_o = (R + \frac{T}{2}) \cdot \theta$$

- The distance traveled by the outer wheel (S_o) along the curved path is calculated as the product of the turning radius (R), plus half of the track width ($\frac{T}{2}$), and the heading angle (θ).

6. Angular velocity of the individual wheels for the inner wheel:

$$\omega_i = \frac{S_i}{r_w \cdot t}$$

- The angular velocity of the individual wheels for the inner wheel (ω_i) is calculated by dividing the distance traveled by the inner wheel (S_i) by the product of the wheel radius (r_w) and the time (t).

7. Angular velocity of the individual wheels for the outer wheel:

$$\omega_o = \frac{S_o}{r_w \cdot t}$$

- The angular velocity of the individual wheels for the outer wheel (ω_o) is calculated by dividing the distance traveled by the outer wheel (S_o) by the product of the wheel radius (r_w) and the time (t).

3.2 Usage

These formulas will be used for analyzing the behavior of vehicles during turns. This will be used to make a robot model for the simulation of the turning movement of the car, different parameters, such as wheelbase (L), track width (T), velocity (v), time (t), and steering angle (θ), affect the turning radius and wheel angles and hence will be determined in this model.

4 Process Model

4.1 Class Diagram

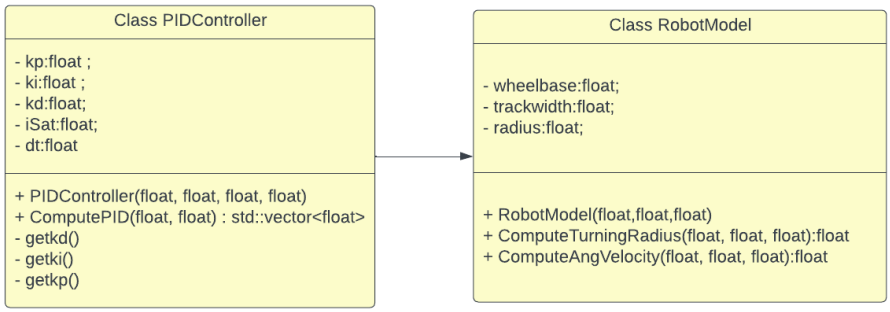

Figure 2: Class diagram

4.2 Activity Diagram

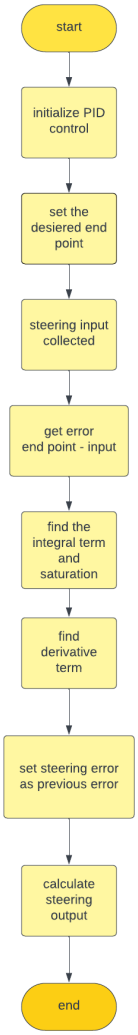

Figure 3: Activity diagram for PID controller

5 Algorithm and methodology

- The target heading and velocity of the robot are received from the user
- Then the internal parameters for the controller like proportional gain, integral gain, and derivative gain are defined
- The error for the controller is initialized as the difference between the current position and desired position of the robot
- The steering angle and the individual wheel velocities are calculated and the position of the robot is updated to the next point in the robot's calculated trajectory
- The error is recalculated again based on the value of the position of the robot

- The above process is continued until the current position of the robot is equal to the desired position of the robot.
- The final position and orientation of the robot is output and also the drive wheel velocities.

6 Tools and resources

The software development team is using C++ 17 with Microsoft Visual studio code IDE configured with clangd, cp- pcheck, valgrind and cpplint. This is a scalable and self-contained API is developed using modern software practices along with an Object Oriented programming approach. The Deoxygen tool is used for the documentation of the software program at the developer level and GitHub is used to maintain all versions of software changes. All builds will be verified using TravisCI and Coveralls with respective unit test cases using the Google Test framework. The dependencies and libraries used for software development are:

- Ubuntu 22
- CMake
- GitHub
- Valgrind

7 Assumptions

- All the parameters of the vehicle are known including initial position and orientation
- Desired position, orientation and velocity of the vehicle are taken as inputs from user
- The maximum steering angle is 45 degrees
- There is no slippage from the tyres of the wheel with respect to the ground
- Each wheel rotates about a vertical axis at its center,
- The steering angles of the wheels are constant throughout the steering maneuver

8 Potential risks and mitigation

After attaining stability, the orientation error fluctuates between threshold values. Estimating the increment/decrement steps to calculate position and orientation errors at specific instants in the required trajectory was difficult. The behaviour of system changes drastically if random steps are considered. An effective way can be implemented to find out the appropriate increment/decrement steps. For a set of inputs, the system doesn't behave as expected. This is observed because the minimization of position error and orientation error are dependent on each other in this algorithm. If we can find a way to make them independent, stability and consistency in the behaviour of the system can be improved.

9 Final deliverables

- The C++ module to control the Ackermann kinematic module
- Overview of the project including timeline, risks, mitigations and UML diagrams
- Link to the GitHub repository with a detailed ReadMe file

- Agile Iterative Process documentation and code coverage with TravisCI and Coveralls
- Memory leaks check and profiling using Valgrind
- Code documentation using Deoxygen tool