Mini Smart Car Hardware Design

RUTGERS
WINLAB | Wireless Information Network Laboratory

Brandon Cheng(UG), Michael Mogilevsky(UG), Aamay Puntambekar(UG) - 2022

Overview

At WINLAB, there are many projects that involve self-driving cars in the orbit smart city. In the past, these projects utilized remote control cars and smart car kits, which lack accuracy in modeling the functionality of a real car.

In our project, we hope to provide greater simulation accuracy with our design for a ~1/15 scale platform for miniature smart cars such that they can be repeatedly tested in the orbit smart city environment. This places an necessity for repeatability and precision using sensor feedback while adhering to the mechanics of a real car.

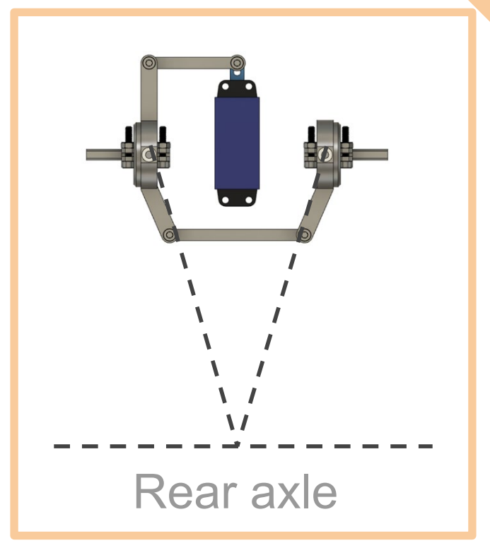
Our design includes many such mechanisms, such as Ackerman steering and a single drive motor running the back wheels through a differential gearing system and drive train, as well as sensors to calculate odometry data.

Our project involves:

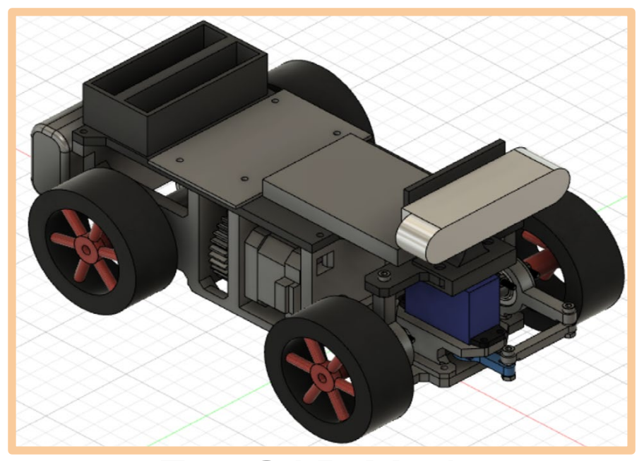
- 3D modeling and printing of as many parts as possible
- Prototyping electrical systems and hardware
- Interfacing between hardware components through different means of communication to control the hardware as well as collect sensor information
- Cross-functional collaboration with teams within the intersection group (Remote Control, Autonomous Infrastrucutre, and Multi-Cam Teams)

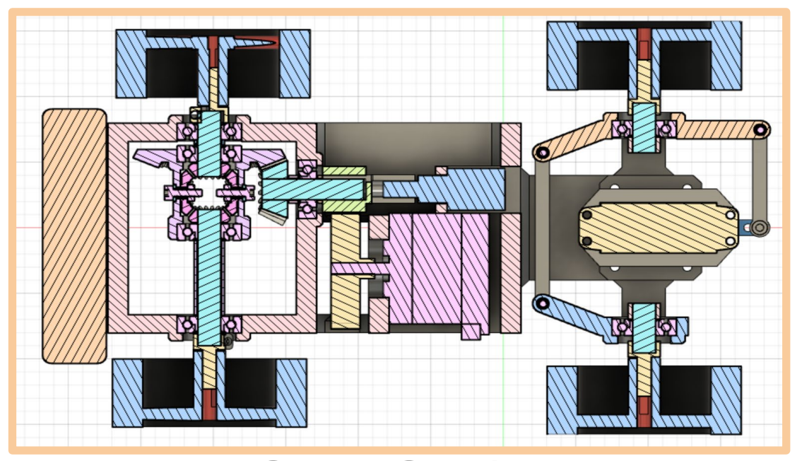
Ackerman Steering

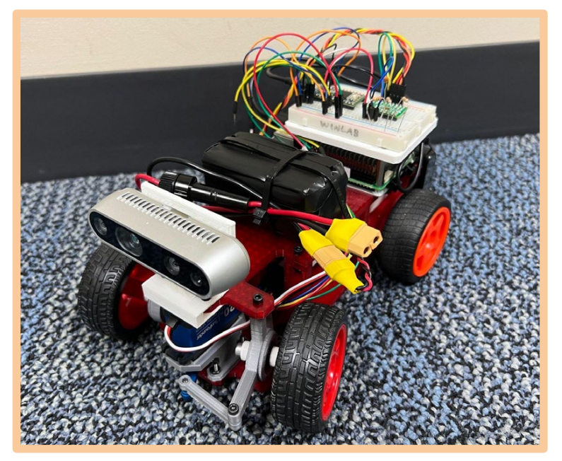
Another complication with turning is that the wheels must turn different amounts. Since the outer wheels travel a longer distance, their arc is larger with a shallower angle than the inner wheels. This is achieved by an Ackerman Steering linkage.

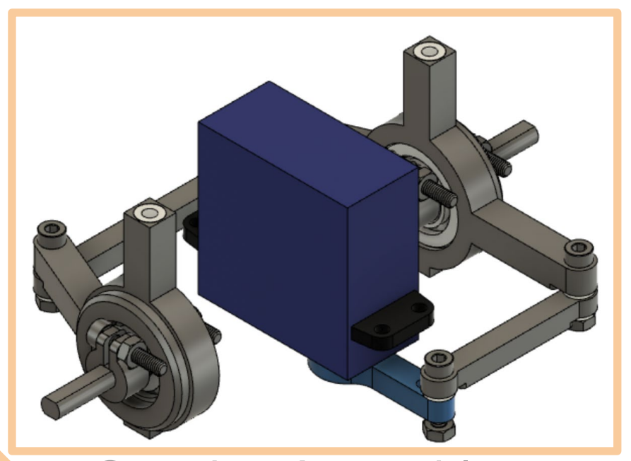


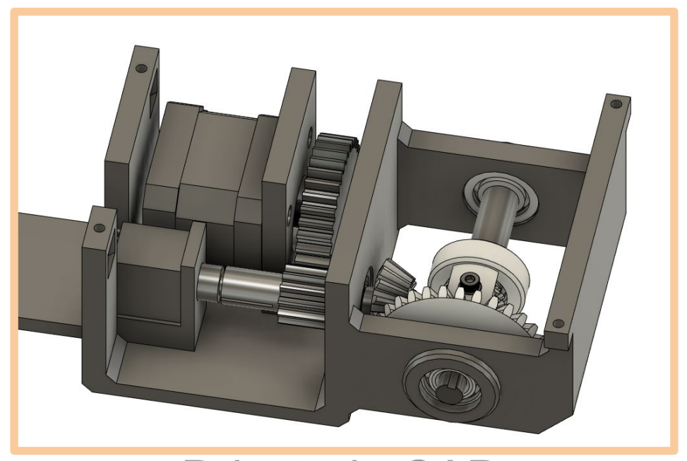
Final Design

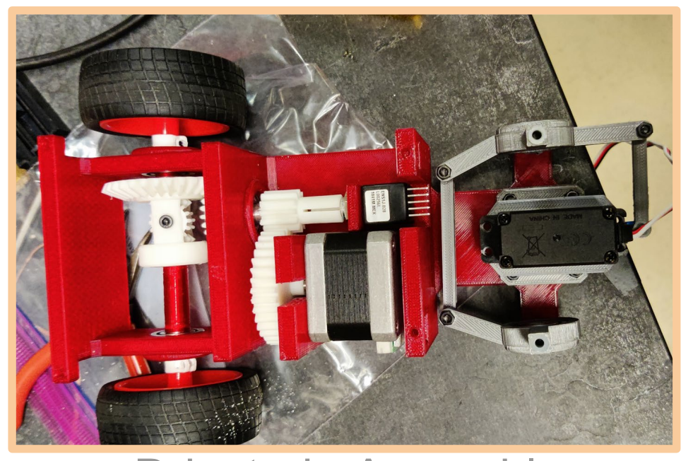

Full CAD Model


Cross Section


Full Assembly


Steering Assembly


Drivetrain CAD


Drivetrain Assembly

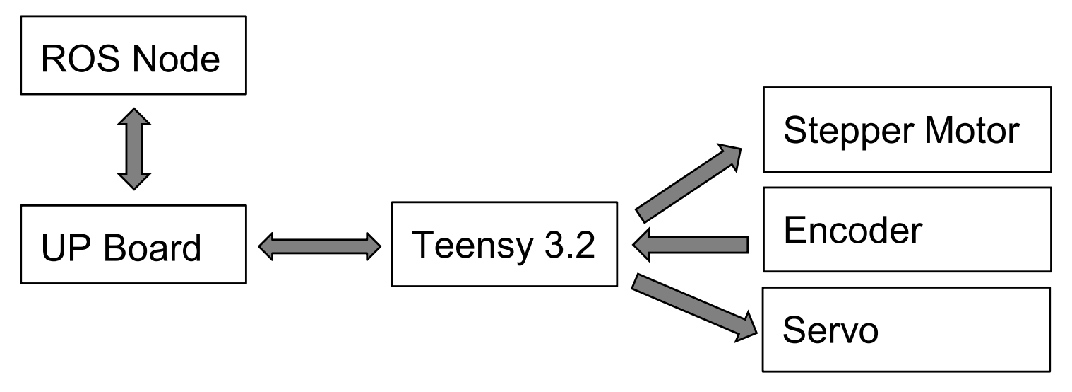
Software

Software Environments

Teensy: Arduino software
UP Board: Ubuntu 20.04, Python, ROS Noetic

Teensy 3.2

- Reads input serial data
- Runs motor and servo
- Sends encoder data through serial



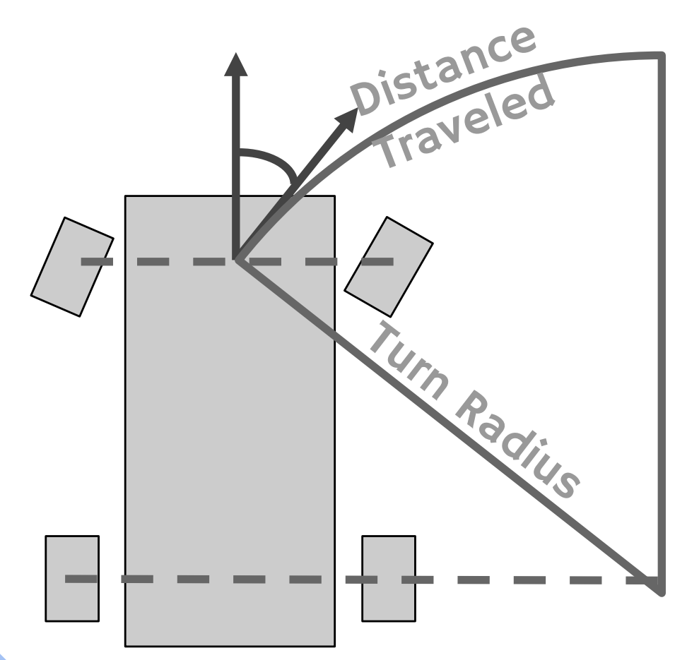
Input Serial Data: Reads 32-bit encoded integer from the UP Board. Decodes the first 16 bits representing RPM (rotations per minute) and the others representing Servo Angle using Bit Shift Operations

Example:

0000000000011110 0	0000000101101000
Servo Angle = 30	RPM = 360

UP Board

- Reads input RPM, desired Servo Angle from ROS Node and sends to Teensy
- Reads the cumulative number of encoder ticks from Teensy
- Calculates the average turning angle of wheels, and then the curvature of the car
- Executes Odometry calculations
- Publishes position and angle to ROS



Odometry

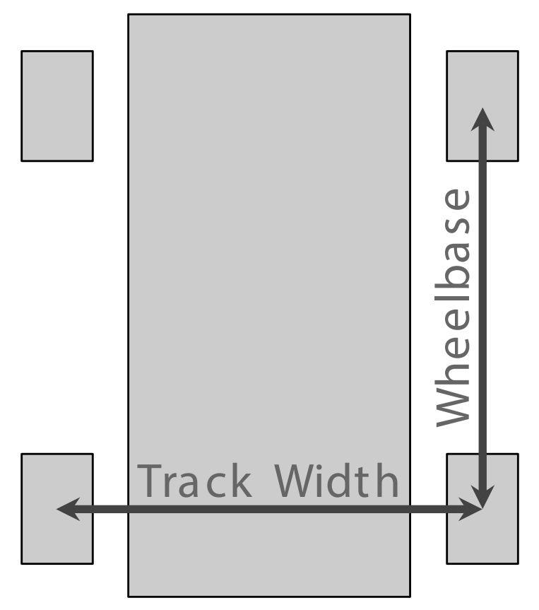
The car constantly calculates its position using the sensor feedback and the servo angle. These are used to determine the distance traveled on an arc, and position is updated accordingly.

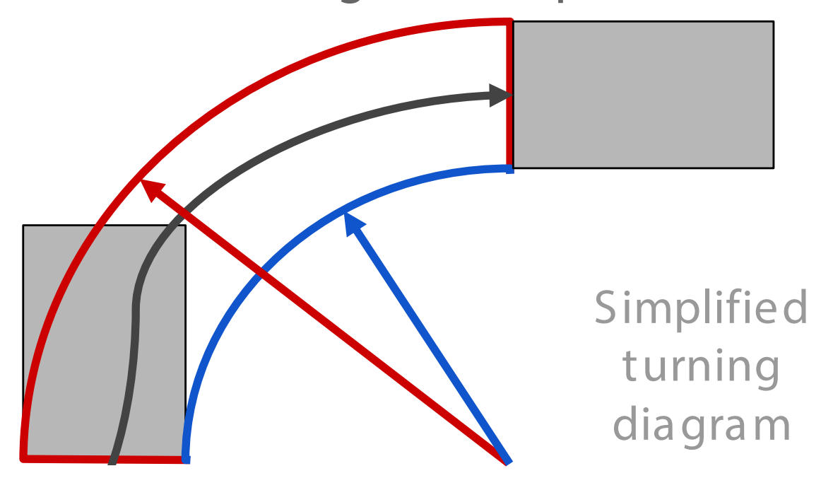
Design Principles

To emulate the behavior of a real car as closely as possible, there are several principles that must be incorporated into the design.

Overall Form Factor

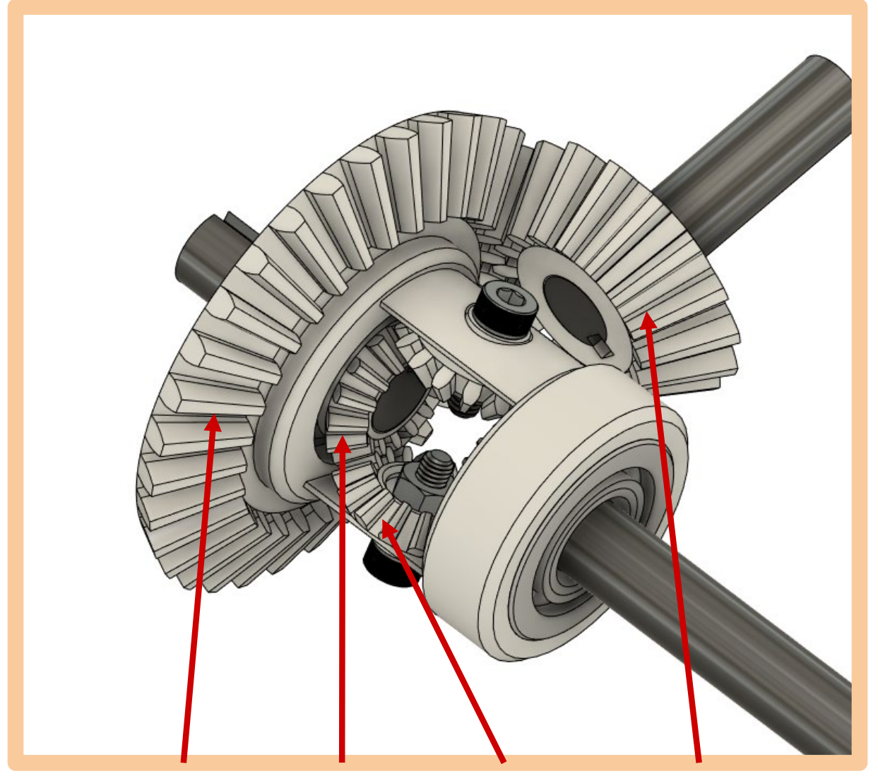
To conform to the size and shape a car would have in the Orbit Smart City environment, the goal was to design the dimensions to be ~1/15 the size of an average car. In our final design, the wheelbase is 150 mm and the track width is 125 mm. Due to some limitations with receiving parts, this makes our car slightly wider than an average car to scale. In the future the track width can be lowered using the required hardware.

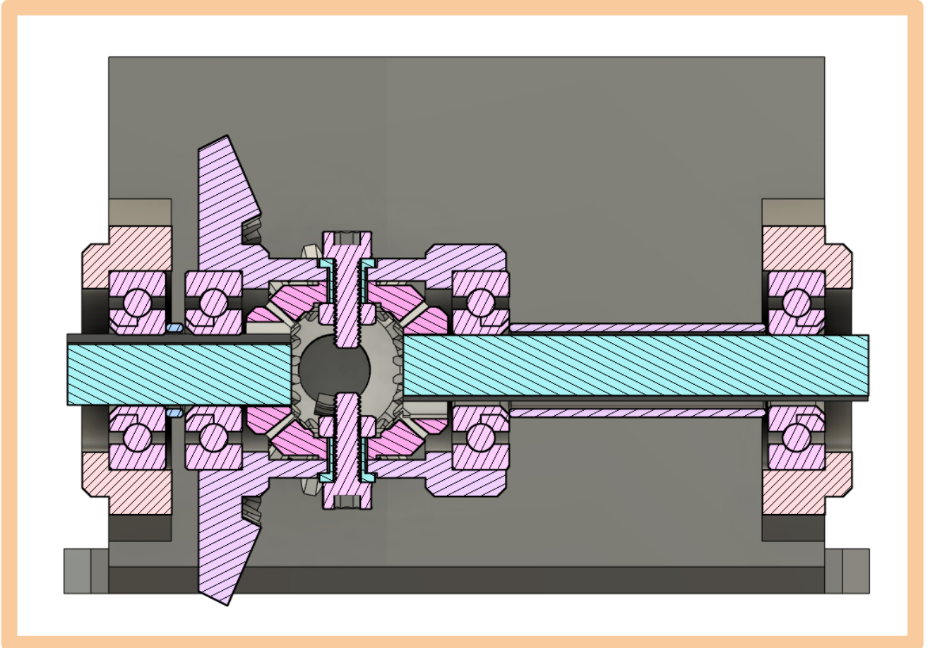



Simplified turning diagram

Differential Drive

In a back wheel drive car, when the car executes a turn, the outside wheel (red) must travel further than the inside wheel (blue) since its arc radius is larger. To solve this, our design incorporates a differential gearing system, which allows the motor to drive the two sides at different speeds since they are loosely coupled.




CAD Cross Section of differential

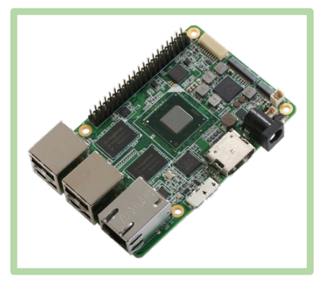
Differential Mechanics

The drive gear is the input which turns the bevel gear. When the car is driving straight, the spider gears don't rotate. This causes the side gears and bevel gear to rotate equivalent amounts. When the car turns via the front steering, one side is forced to turn more than the other. This causes the spider gears to rotate, which adds rotational speed to one side and subtracts from the other.

Electronics

Portable Phone Charger:
Supplies 5V power to the UP Board, which then supplies 5V power to the Teensy and other components through a serial port.

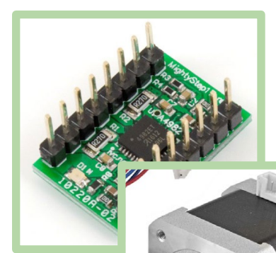
Up Board:
A Single Board Computer that communicates with the teensy microcontroller through serial and can communicate externally through the network



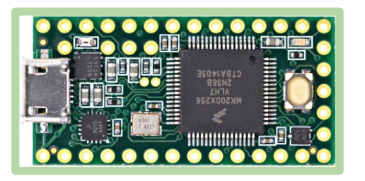
Realsense Camera:
A powerful LIDAR Camera with an onboard IMU, which is connected to the UP board.

LiPo Battery Cells + BMS:
Supplies 4S (16.8V) voltage to power the motor. the Battery Management System (BMS) prevents the LiPo cells from over charging or discharging.

Stepper Motor + Driver:
Powers the drivetrain of the car. The stepper driver generates a signal to drive the motor with a higher voltage supplied by the LiPobatteries.

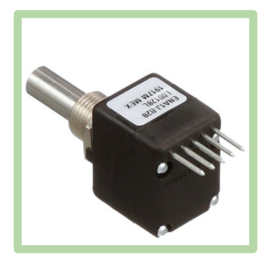


Teensy 3.2:
Microcontroller used to read data from sensors and control hardware in realtime.

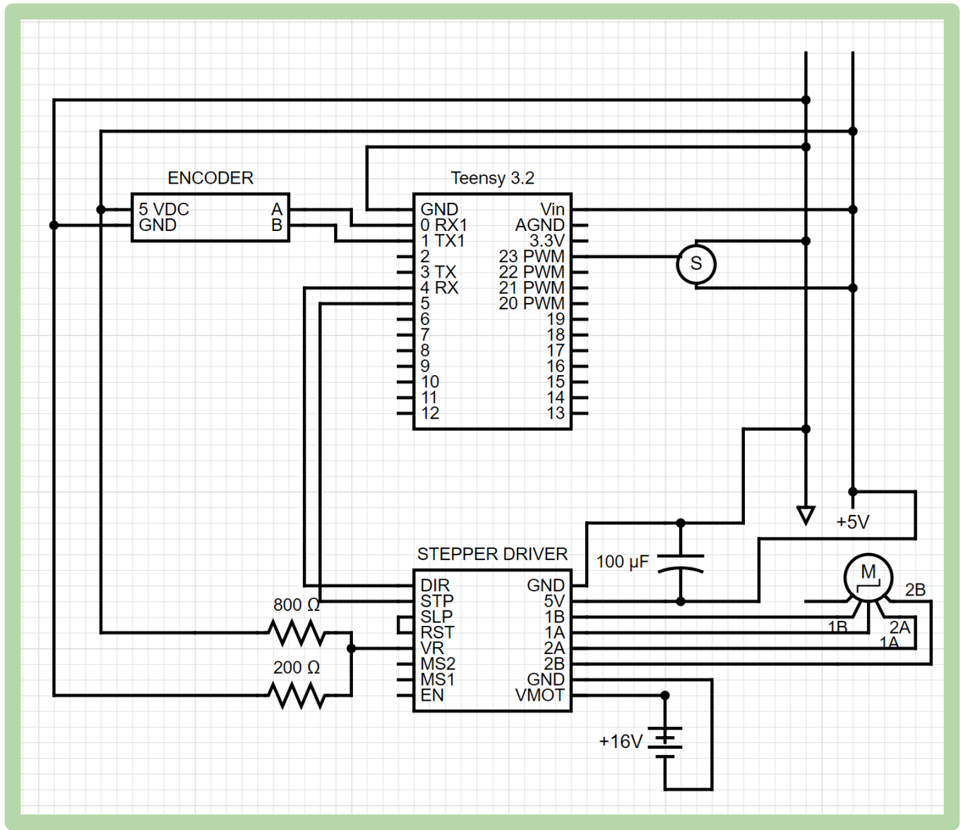


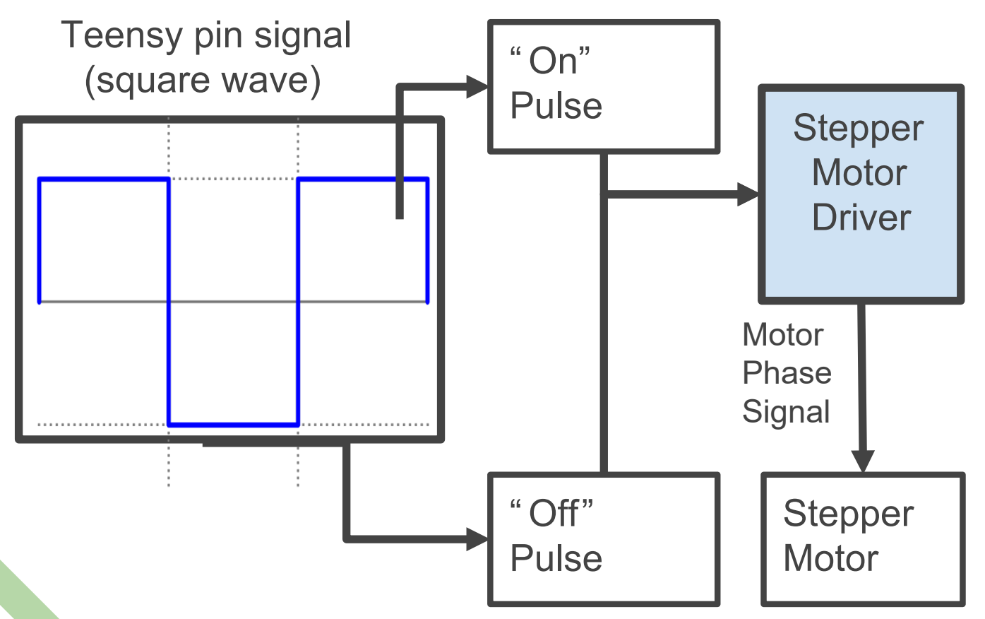
25kg*cm Digital Servo:
Drives the steering mechanism to accurately steer the car based on absolute angular positions.

Bourns EN Optical Encoder:
Provides feedback by generating a square wave. This reading is used to calculate displacement, speed, and position.



Wiring Diagram





Square Wave Signals

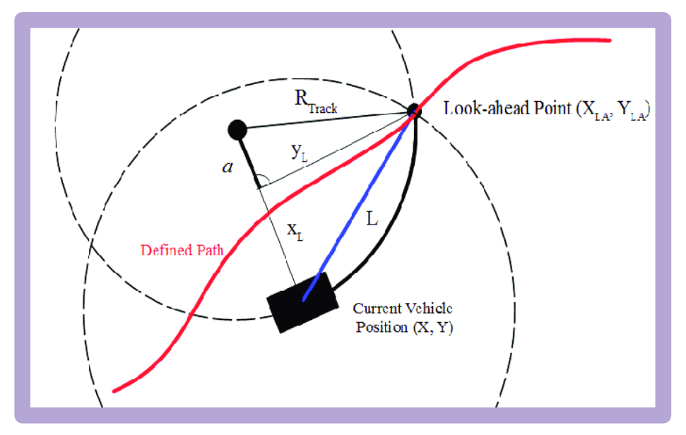
The Teensy controls the stepper motor by sending a square wave to the motor driver, which produces the phase signal necessary to drive the motor by 1 "step" or 1.8 degrees per pulse of the square wave. The speed of the motor can be controlled by the frequency of the square wave.
A similar method is used to interpret the reading from the encoder, which outputs a square wave. The Teensy reads the signal and counts the number of times it turns on and off (one encoder "tick"). The encoder we are using has a resolution of 256 ticks per rotation.

Future Work

With our car we were able to achieve simple control and odometry. In the future, the car's functionality could be expanded through some of these possible future projects. In addition, the hardware tolerances and overall design can be improved.

Spline Path-Following Algorithm

Use real-time Odometry data to work with the Pure Pursuit path following algorithm. This will provide velocity and curvature from the position of the car relative to the path it wants to follow. This will ultimately allow the car to navigate intelligently as it will correct itself when its position differs from the target path.

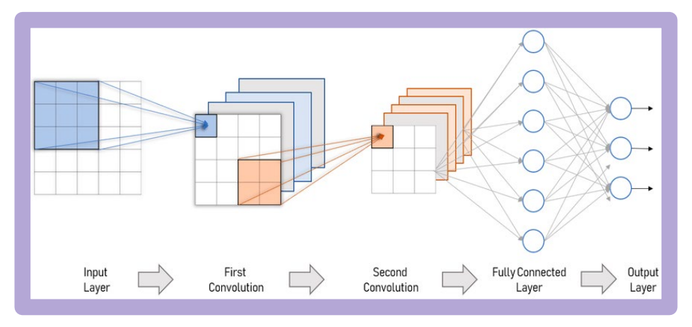


Lidar/Depth Camera Odometry

Implement the RealSense Camera or a lidar sensor with the car's odometry data to calculate its position using depth image information or a lidar sensor.

Neural Network Implementation

Test/train Convolutional Neural Networks on our hardware platform using odometry, camera/Lidar input, and other sensor data. Eventually, implement Lift-Splat-Shoot algorithm.



WINLAB