Autonomous Driving Demonstration and Competition

April 24, 2025

Winter 2025 Group 8

Submitted by:

Lucas Costello, Computer Engineering, CE Program Lead - Computer Subsystem
James Haser, Mechanical Engineering, ME Lead, ME CAD - Mechanical Subsystem
Randa Oroo-Zro, Electrical Engineering, EE PCB Layout/Circuit Design - Abstract, Introduction, Customer Needs and Functions, Standards, Codes, and Patent Research, Conceptual Design (Overview, Electrical Subsystem), Discussion (Technical Discussion, Engineering Standard, Safety, Ethical Considerations, Potential Customers, Societal and Global Impact), and Conclusions and Recommendations
Jon Tezak, Mechanical Engineering, ME Kinematics - Mechanical Subsystem, Discussion (Economic Factors, Reliability, Aesthetics, Information Literacy)
Joseph Tituskin, Electrical Engineering, ECE Lead, EE Circuit Design - Electrical Subsystem

For:

Michael A. Latcha, PhD
Mohamed A. Zohdy, PhD

ME 4999 / ECE 4999 Senior Design Project

Abstract

This project presents the development of a fully autonomous line-following vehicle designed for the Winter 2025 Autonomous Driving Demonstration and Competition. The vehicle integrates mechanical, electrical, and software subsystems to navigate a course of blue tape while adhering to strict performance, safety, and dimensional constraints. Key features include a PixyCam 2 smart vision sensor for line and intersection detection, an ultrasonic sensor for traffic and obstacle avoidance, and an Arduino Mega microcontroller to coordinate sensor data and vehicle behavior. The compact 3D-printed chassis, powered by rechargeable batteries, supports a two-wheel locomotion system with a bumper switch for collision detection. The vehicle is programmed to stop at intersections, yield to other vehicles, and maintain a minimum 200 mm following distance. Emphasizing reliability, affordability, and modular design, the system meets all engineering requirements for functionality, visibility, and safety.

Table of Contents

Introduction

The project for the Winter 2025 Autonomous Driving Demonstration and Competition focuses on designing and building a fully autonomous line-following vehicle. The goal is to create a reliable, functional, and affordable vehicle that meets all the competition rules. The vehicle needs to navigate a course using a 1-inch-wide blue tape line, stop and yield at four different types of intersections, and keep a minimum 200-millimeter distance from obstacles and other vehicles.

To meet these objectives, the team focused on prioritizing customer needs and functions as per given assignment material, necessary engineering requirements and their confirmation processes, and the design of the vehicle that takes these functions, requirements, and confirmations into account. Some examples of required functionality include a compact footprint (maximum 25 cm x 25 cm), running at speeds between 100 mm/s and 200 mm/s, operating for at least 120 minutes without interruption, and being able to handle tight turns and different course conditions. Additional requirements include the use of bumper switches for collision safety and LED indicators to meet visibility standards.

The proposed design integrates three subsystems–mechanical, electrical and software–to make everything work cohesively. The mechanical subsystem includes a 3D-printed chassis constructed from ABS plastic for lightweight durability and a two-wheeled locomotion system to enhance maneuverability. The electrical subsystem includes a PixyCam 2 for line following, an ultrasonic sensor for obstacle avoidance, and LEDs & a limit switch for traffic and collision avoidance. These components are mounted on a prototype shield board to organize circuitry and minimize internal clutter. The software subsystem processes data from the sensors to navigate the vehicle, stop at intersections, and follow traffic rules using the Arduino Mega Microcontroller. This design is made to meet all the competition requirements while being easy to adjust if any challenges arise. The team's approach ensures the vehicle will be ready to perform to specification, and perform well in competition.

Customer Needs and Functions

The process of identifying customer needs and functional requirements for the autonomous vehicle project involved several steps to ensure that the design aligns with competition objectives and real-world applications. The team began by reviewing the project description and competition rules provided by the sponsors/instructors, which highlighted key requirements such as reliability, line tracing, traffic compatibility, and maneuverability were identified as the highest priorities due to their importance in ensuring the vehicle's success in the competition. Affordability was also heavily weighted to reflect the strict budget constraints. To clarify design requirements and functional expectations.

The team reached out to the instructors via email. Their feedback provided valuable insights that helped refine and adjust the Customer Needs and Functions worksheet, ensuring that priorities were appropriately weighted. The updated Customer Needs and Functions worksheet prioritizes the design elements necessary to meet competition requirements and real-world challenges.

Needs / Functions	Priority	Notes
Footprint	3	The vehicle must have a compact design, with a maximum allowable footprint of 25 cm x 25 cm.
Reliability	10	High reliability is essential for autonomous functionality during the competition, ensuring the vehicle performs consistently across various conditions.
Affordable	9	The project must remain within the $300 budget constraint while maintaining the required functionality.
Line Tracing	10	The vehicle must accurately follow a 1-inch wide blue tape line.
Traffic Compatibility	10	Sensors must ensure the vehicle maintains a safe 200 mm distance from obstacles and yields appropriately at intersections.
Maneuverability	10	The vehicle must have excellent maneuverability to navigate tight corners, curves, and varying surface conditions.
Speed	2	minimal impact on performance.
Bumper Switch	5	Provides safety by detecting collisions.

Table 1: Customer Needs and Functions

Standards, Codes, and Patent Research

The design process for the autonomous vehicle involved researching similar technologies, identifying relevant safety standards and patent considerations to ensure functionality, compliance and reliability. The following standards and guidelines were reviewed and incorporated into the project:

- ISO 13849: A standard that addresses the safety of machinery and equipment, including components of automated vehicles, by evaluating risks and ensuring adequate safety measures are in place for each component. [1]
- ISO 22839: This standard addresses testing methodologies and validation of autonomous driving systems, focusing on the performance and safety of sensors, control systems, and algorithms. [2]
- SAE J3016: Developed by the Society of Automotive Engineers (SAE), this standard defines the levels of driving automation (Levels 0 to 5), providing a classification system for the degree of automation in a vehicle. It helps to categorize AVs based on their capabilities and operational characteristics. [3]
- IPC-A-610: This standard is essential for ensuring the quality and reliability of electrical components and proper assembly techniques, covering soldering, wiring, and PCB construction. [4]
- UL 4600: This standard provides comprehensive guidance for ensuring the safety of autonomous systems, such as self-driving vehicles, drones, and robots. [5]

<u>**Engineering Requirements and Specifications**</u>

Development of a list of Engineering Requirements and Specifications was completed over the course of several meetings with every team member to refine both Customer Needs and Functions, as well as tasks gathered from the assignment worksheet to a list of practical goals that can be completed during development. Each member of the team brought up concerns and challenges regarding development pertaining to their area of expertise, these being electrical, mechanical, and computer engineering. Potential design solutions were proposed by all members until one had resonated with each member. This process dictated the target solutions shown on our Design Requirements spreadsheet. After meeting with the instructors, the list was updated to include their feedback. This made the list more complete and focused by removing goals that could not be tested or measured.

The Engineering Requirements and Specifications list was organized in a way where different requirements were separated into different categories or areas of design. These categories did not define requirements by subsystem, but by aspect of design, although they did help in the organization of subsystem requirements. These categories are Structure, Motion, Line-Tracing, Traffic, Power, and Cost. Structure refers to the chassis and physical makeup of the device, Motion pertains to how it traverses its environment, Line-Tracing regards the devices function of following a tape line on the ground, Traffic refers to how the car senses its environment and acts accordingly, Power pertains to input power and functionality, and Cost establishes a target development budget. Both mechanical and electrical subsystems are derived from a number of these categories. For example, the motion category encapsulates mechanical targets and designates a desired vehicle speed, but this designation can interfere with traffic conditions, which is more a part of the electrical subsystem. As such, some requirements attempt to address these issues before they arise, and as such, the team had decided that some sort of distance tracking that can adjust speed based on the speed of a preceding vehicle on a track would be necessary to ensure functionality.

As development progressed, testing began on each requirement. The team completed validation for the vehicle’s footprint, input power, and turn signal construction. These early completions helped confirm that the foundational elements of the design met their respective specifications. Other requirements—such as operating speed, environmental adaptability, and intersection detection—remain in progress as integration testing continues.

Category	Engineering Requirements	Importance	Target	Validation Method	Status
Structure	Footprint	High	< 25cm x 25cm	CAD analysis, Inspection	Complete
Structure	Motor Mount	Med	Withstand 15 Nmm	CAE Analysis	Complete
Motion	Turning Radius	High	0 < x < 250 mm	Estimate through analysis	Complete
Motion	Operating Speed	High	100-200 mm/s	Physical Validation	Complete
Line Tracing	Line Following Tolerance	High	Follow 1" wide painter's tape +/- 10 mm	Physical Validation	Complete
Line Tracing	Intersection and Turn Detection	Med	Stop 15 +/- 5 mm prior to stop line	Physical Validation	Complete
Traffic	Following Distance	Low	> 200 mm	Programming, Physical Validation	Complete
Traffic	Distance Sensor	High	> 300 mm	Component Datasheet	Complete
Traffic	Turn Signal Construction	High	< 25cm	CAD analysis, Inspection	Complete
Power	Input Power	Med	12 V	Microprocessor Power Ratings	Complete
Power	Operating life	High	> 2hr	Estimate through analysis	Complete
Cost	Design project cost	High	< $300	BOM	Complete

Table 2: Engineering Requirements and Validation

Conceptual Design

Overview

The proposed design is for a fully autonomous line-following vehicle. By integrating mechanical, electrical and software subsystems, the vehicle achieves reliability, affordability, and safety while meeting all competition requirements. The focus is on modularity and functionality to make sure the design is robust and adaptable to any challenges faced during the competition.

The mechanical subsystem provides the structural foundation for the vehicle, balancing compactness, stability, and durability. The chassis is built using 3D-printed ABS plastic due to its lightweight and cost-effective properties. The design fits within the competition's 25cm x 25cm footprint limit and securely contains all components. A two-wheeled locomotion system is chosen to enhance maneuverability, allowing the vehicle to handle tight corners and different surface conditions effectively. Plus, the front of the vehicle has a bumper switch to ensure safety and protect the vehicle from damage.

The electrical subsystem includes components such as the battery, an Arduino Mega microcontroller, motors, and sensors, which are essential for powering and controlling the vehicle's movement. A camera with a neural network detects and follows a 1-inch wide blue tape, while an ultrasonic sensor handles traffic detection, maintaining a safe 200 mm distance from other vehicles. The power system uses rechargeable batteries that can sustain continuous operation for over 120 minutes. LED indicators are included at the front and rear of the vehicle to meet visibility requirements. All electrical components are mounted on a soldered perfboard / prototype shield board to make sure they are durable and reliable.

The software subsystem is responsible for controlling and managing the vehicle's autonomous functions. The system consists mainly of an Arduino Mega microcontroller programmed in C++ to process real-time sensor data for navigation tasks, such as line following, stopping at intersections, yielding to other vehicles and resuming operation. In addition, a PixyCam 2 is mounted using a pan/tilt kit to obtain and identify line-following and directional device signatures.

Challenges may include ensuring stable communication between subsystems and maintaining consistent performance over long periods. These challenges will be addressed through extensive testing and thorough validation of all components before final assembly. With these measures, the vehicle is well-prepared to succeed in the competition.

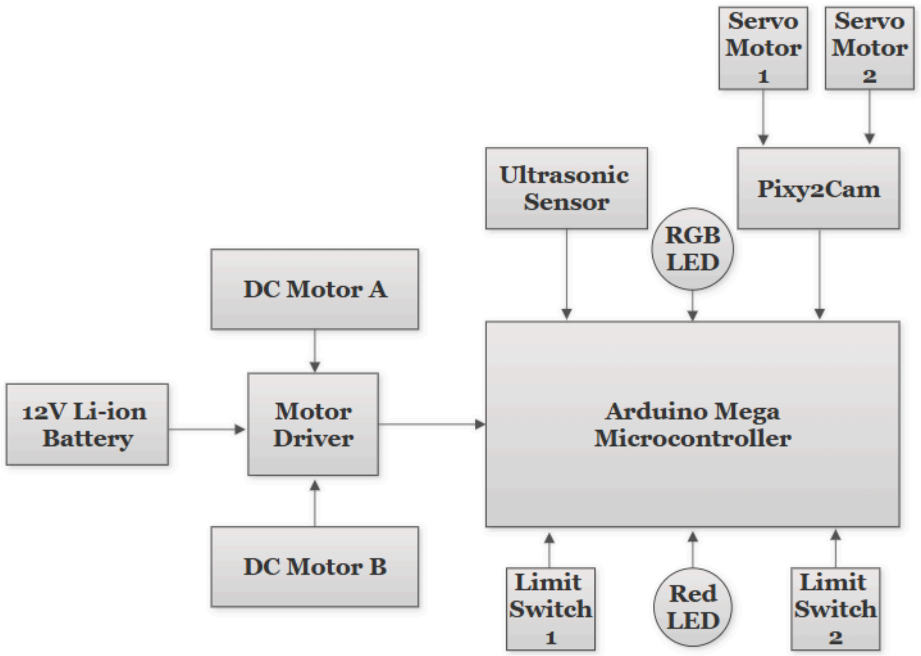


Figure 1: Block Diagram

The diagram above illustrates the overall hardware architecture of the autonomous vehicle. The Arduino Mega microcontroller coordinates data from the Pixy2Cam, ultrasonic sensor, and limit switches to control the motor driver, LEDs, and other peripherals, enabling full autonomous navigation.

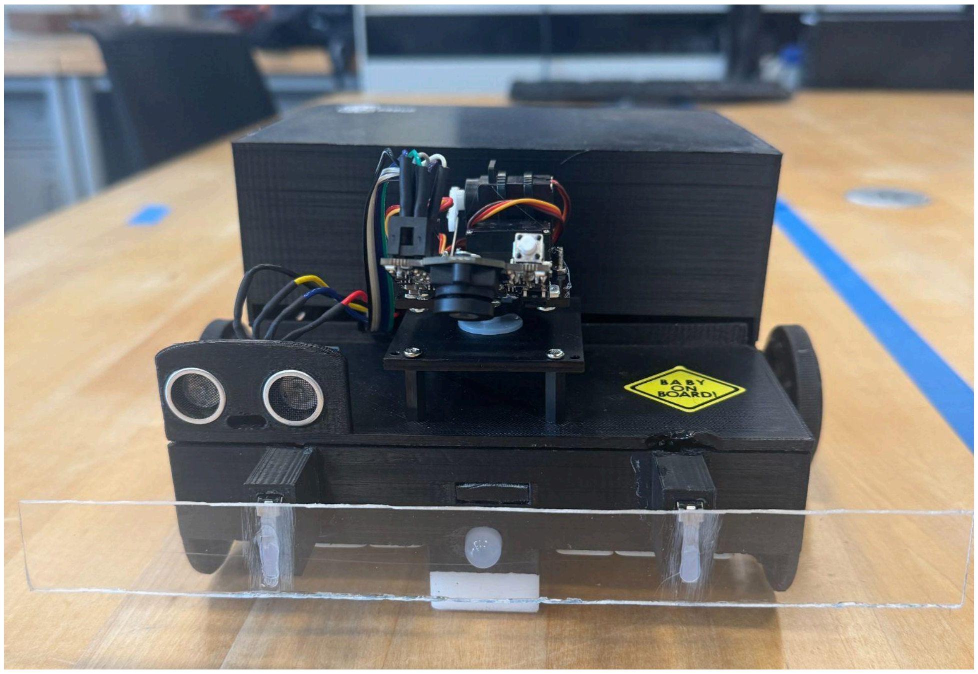


Figure 2: Final Physical Assembly of the Autonomous Vehicle

<u>Mechanical Subsystem</u>

For the overall mechanical design, two main elements were considered: the chassis and the method of motion. The chassis was to efficiently and cohesively house all components, while the method of motion was to reliably propel the vehicle around the course. In addition, both of these must be accomplished while staying within the allocated budget. Taking these factors into consideration, a two-wheeled design was chosen, supported by a 3D-printed chassis made of ABS. This final design is pictured below:

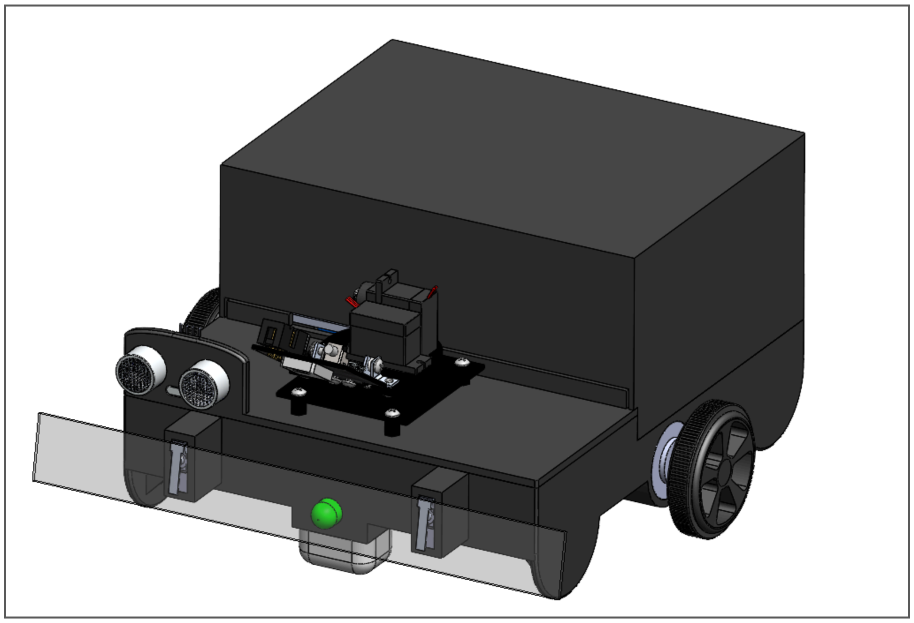


Figure 3: Final CAD model of the mechanical subsystem

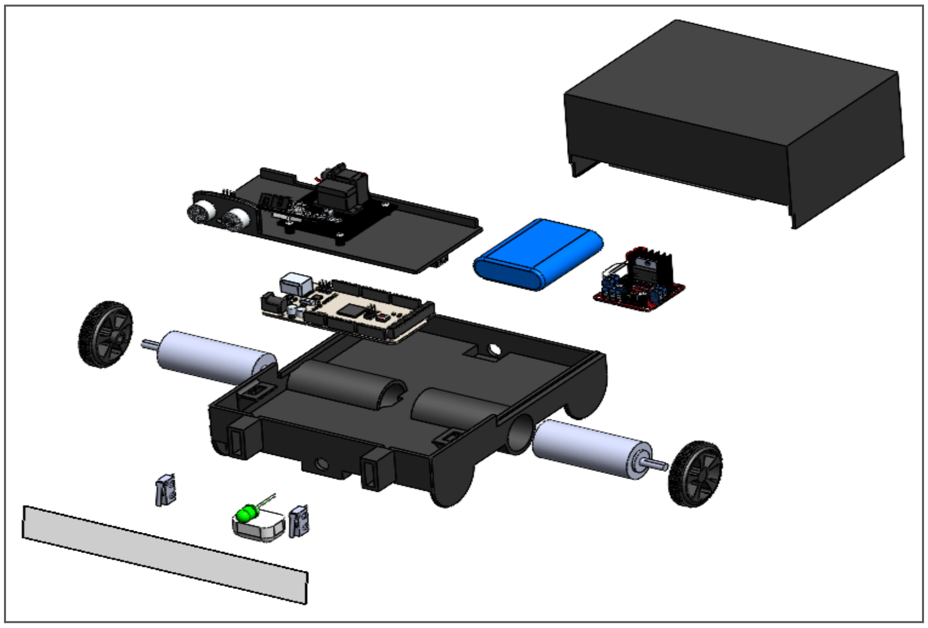


Figure 4: Final CAD model exploded view of the mechanical subsystem

This design meets all relevant specifications listed in the project description:

- Footprint of 214 mm x 220 mm (max 250 mm x 250 mm)
- 10-mm LEDs on front and back placed 25 mm from floor
- Wheels and motors for locomotion
- Front bumper switch spanning the total width of the vehicle

In addition, all necessary components for autonomy, line tracing, and power are included.

While developing this design, both the material and its method of manufacture had to be considered. With the latter having a direct impact on the former, it was crucial to determine methodology before the material itself. By the metrics of cost, weight, dimensional stability, and sustainability, the following decision matrix was created:

Material Manufacturing Method					
Criteria	**Weight**	**3D Printing**	**Waterjet Plastic Plates**	**CNC Plastic Plates**	**Laser Cut Plates**
Cost	3	+6	+3	+3	+3
Weight	5	+10	+5	+5	+5
Dimensional Stability	1	+2	+1	+2	+2
Eco-Friendly / Waste	2	-2	-4	-4	-4
	Total	+16	+5	+6	+6

Table 3: Decision Matrix for Material Manufacturing Method

As shown in Table 3, 3D printing was demonstrated to be the best fit for the purposes of this project. This comes in part due to the nature of 3D printing being an additive manufacturing process, as opposed to traditional subtractive processes. The considerable reduction in waste material also aids in cost and weight reduction, since infill, or the amount of material between walls, is a modifiable property in 3D printing [7], [8].

In choosing the material, several common 3D printing filaments were compared against traditional aluminum, which is a popular material in the application of small vehicles. This was done in part to justify a custom fabricated chassis. Considering cost, weight, and material yield strength (durability), the following decision matrix was created:

Material Choice								
Criteria	**Weight**	**Alumi-num**	**PLA**	**ABS Plastic**	**Carbon Fiber PLA**	**PC-ABS**	**Nylon 6**	**Nylon Carbon Fiber**
Cost	4	0	-4	-4	-4	-8	-8	-8
Weight	3	0	+0	+6	+3	+3	+6	+3
Durability	2	0	-4	-2	-2	-2	-2	-2
		Total	-8	+0	-3	-7	-4	-7

Table 4: Decision Matrix for Material Choice [9] - [15]

As shown in Table 4, ABS plastic was demonstrated to match aluminum overall, with its higher cost and lower durability compensated by lower overall weight. Because of this, ABS plastic was chosen for the initial design.

Choosing how the vehicle moves is a crucial step, as mode of locomotion affects almost every other aspect of the vehicle. Not only was mode of locomotion important, but also the quantity of whatever type was chosen (for example, a 3-legged vehicle vs. 6 or 8-legged). These decisions are documented in the following matrices:

Locomotion Type				
Criteria	**Weight**	**Wheels**	**Tracks**	**Legs**
Cost	4	1	-2	-1
Weight	5	1	1	1
Footprint	3	2	-2	2
Complexity	1	2	-2	-2
Speed	2	2	1	1
	Total	21	-9	7

Table 5: Decision Matrix for Locomotion Types

Quantity of Best Locomotion Type				
Criteria	**Weight**	**2x**	**3x**	**4x**
Cost	4	1	0	-1
Weight	5	1	0	-1
Footprint	3	1	0	-1
Complexity	1	2	1	2
	Total	14	1	-10

Table 6: Decision Matrix for Quantity of Best Locomotion Type

As shown in Tables 5 and 6, the decision was ultimately made to use two wheels for locomotion. Wheels were chosen for their lightness, simplicity, and capability, while two wheels furnished the least weight and complexity. With a two-wheeled design, drivetrain complexity is minimized, however, this design necessitates the use of a support mechanism for stability. The matrix for choosing a method of support is shown below:

Support Mechanism				
Criteria	**Weight**	**Felt Tape**	**Ball Bearing**	**Micro Controlled**
Cost	4	8	4	-4
Weight	5	10	5	10
Footprint	3	3	3	6
Complexity	1	2	1	-2
	Total	23	13	10

Table 7: Decision Matrix for Support Mechanism

In Table 7, felt tape was initially shown to be the optimal choice for the design, due to its low cost and weight. However, during the testing process of the vehicle, excessive friction and wear were observed, which had not initially been considered in the group's decision of a support mechanism. Upon this observation, the decision was reconsidered while including a mix of old and new options and using a revised list of criteria.

Support Mechanism (revised)				
Criteria	**Weight**	**Felt Tape**	**PTFE**	**Ball Bearing**
Cost	1	0	-1	-2
Weight	3	0	-3	-6
Footprint	2	0	4	4
Friction	3	0	6	3
Complexity	2	0	-2	-4
	Total	0	4	-5

Table 8: Revised Decision Matrix for Support Mechanism

With the inclusion of friction as a criterion, PTFE was ultimately chosen as the support mechanism. Additionally, upon recommendation of the instructor, the four-corner design was retired in favor of two support mechanisms centered on the front and back of the vehicle.

Aside from the method of support, one more factor to consider in the vehicle's interaction with the floor was the interaction between the locomotive devices and the floor. With wheels being the chosen method of locomotion, a few methods of wheel-floor interaction were considered. The initial decision matrix is shown here:

Wheel/Floor Interaction Method				
Criteria	**Weight**	**Raw ABS**	**Electrical Tape**	**Rubber Tire**
Cost	1	2	1	-2
Traction	3	-6	3	3
Complexity	2	4	2	-2
Durability	2	2	-2	2
	Total	-2	2	1

Table 9: Decision matrix for interaction between the wheel and floor.

From Table 9, it was initially decided to use electrical tape as a wheel cover. However, throughout the development process, traction proved to be an issue as well as debris buildup from the floor surface. With these issues, as well as sub-optimal aesthetics, electrical tape and rubber tires were re-evaluated as follows:

Wheel/Floor Interaction Method (Revised)			
Criteria	**Weight**	**Electrical Tape**	**Rubber Tire**
Cost	1	0	-1
Traction	3	0	6
Complexity	2	0	-2
Durability	2	0	4
	Total	0	7

Table 10: Revised decision matrix for wheel/floor interaction method.

As shown in Table 10, rubber tires were chosen as a replacement interaction method. This comes from vastly increased traction and durability in comparison to the initial method of electrical tape.

With the mode of locomotion decided, a torque calculation was required to size the drive motors appropriately. Starting with a free-body diagram,

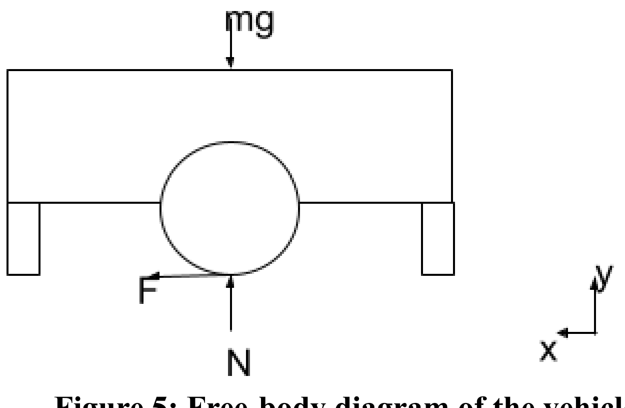


Figure 5: Free-body diagram of the vehicle.

From a force balance, taking the mass as .973 kg, the normal force was calculated to be 9.535 N. For the driving force F, using Newton's second law:

$$F = ma$$

While the mass m is known, the acceleration must be calculated. Using a target acceleration time of 0.1 seconds, we can find the acceleration using a target velocity of 0.180 meters per second and the following equation:

$$v = \int a dt$$

Solving this integral and substituting in values, we obtain

$$0.180 = 0.1a + v_0$$

Since the initial velocity is 0, we can cancel the final term and solve for a, which gives us a required acceleration of 1.8 m/s^2. Substituting a back into Newton's second law, the driving force F was found to be .9 N. We can then use F to find the torque, using the wheel radius of 0.025 meters:

$$T = rFsin\theta$$

With an angle of 90 degrees between the driving force and the wheel radius, we find the required torque to be .0225 Nm. Noting that two drive motors are used in the proposed design, the required torque per motor becomes .0113 Nm. This is an important figure in sizing the motor.

Three main types of electric motor were considered for this application: brushed DC, brushless DC, and stepper motors. To decide which type of motor to use, the following matrix was created:

Electric Motor Type				
Criteria	**Weight**	**DC Brushed**	**DC Brushless**	**Stepper Motor**
Cost	3	3	-3	3
Power	4	4	-4	0
Efficiency	2	-2	2	-2
Operating Speed	1	0	0	-1
	Total	5	-5	0

Table 11: Decision Matrix for Electric Motor Type

Comparing the three motor types against each other in Table 13 showed that brushed DC motors were ideal, due to their cost-effectiveness and high torque [16], [17].

Initially, the group elected to use a 9-volt DC motor sold on Electronix Express. This motor had a torque rating of 44.2 gram-centimeters, which translates to .00433 Nm. Aside from the initial motor's torque not meeting the calculated requirements, the operating speed proved problematic, having a maximum speed of 12,500 revolutions per minute. Because of this, the decision was made to change to a motor with an internal gearbox. The included gearbox allowed for increased torque and greatly reduced operating speed.

Motor Design Change			
Criteria	**Weight**	**Original Motor**	**Motor with Gearbox**
Cost	2	0	-4
Torque	3	0	3
Efficiency	1	0	-1
Operating Speed	4	0	8
	Total	0	6

Table 12: Decision matrix for motor design change

The final chosen motors were obtained from NFP Motors with a gear ratio of 25:1, an operating speed of 160 RPM after reduction, and a torque rating of 1.5 kilogram-centimeters, which translates to .147 Nm. The reduced speed and increased torque allowed for increased controllability and increased ability to achieve required operating speed. The relationship between angular speed of the motor and linear speed of the vehicle was found using the following equation:

$$V = \omega r$$

With a wheel radius of 25 millimeters, and after doing the necessary conversions, the required motor speed was determined to be between 45 and 90 RPM. This calculation was a driving factor in the choosing of our motor.

After implementing the motor into the design or the chassis, a torque simulation was completed to confirm that the motor would not break the chassis during operation. A moment was applied to the interfacing surfaces where the motor contacts the chassis. The load vectors can be seen below.

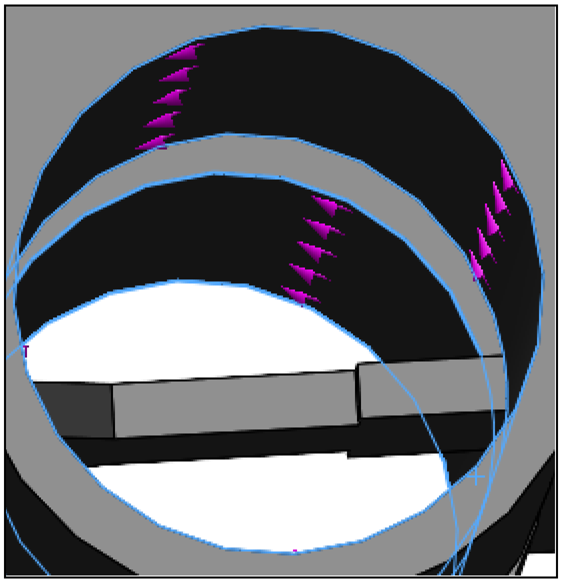


Figure 6: Torque Vectors Applied to Chassis-Motor Interface.

As seen below, the chassis resulted with a factor of safety of 28 along with a colored torque mapping of the distribution of load across the chassis. From this we can see high areas of stress concentration but nothing to be concerned with as the chassis will not yield to the following torque of the motor.

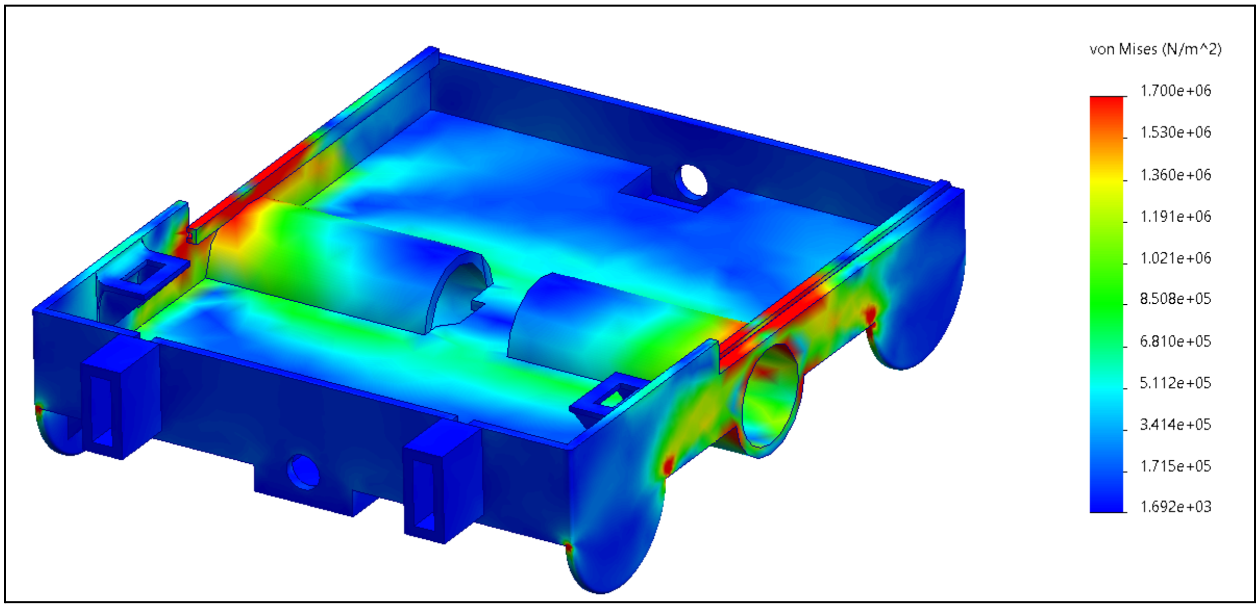


Figure 7: Load Map Across Chassis.

Many of the challenges encountered while developing and testing the vehicle were practical: motor changes were made to satisfy speed requirements, bumper changes were made to ensure proper functionality, the support mechanism was changed to reduce friction, and rubber tires were implemented to improve traction. Other challenges in the subsystem arose from packaging and issues raised during testing of vehicle functions. Namely, a major factor driving design changes was the decision to add pan/tilt functionality to the PixyCam 2. This necessitated reworking the top cover design to accommodate for the mounting points of the Pixy-supplied pan/tilt kit. Another major design change arose from packaging of the PCB: dimensions of the circuit board were unknown at the time of initial design. Consequently, a design change was necessary when PCB size became known.

ME Subsystem Bill of Materials			
Items	Quantity	Cost	Ext. Cost
ABS Filament	0.315 kg (w/ 20% infill)	$21.99/kg	$6.93
Electric Motor	2	$15	$30
Bumper Spring	2	$2.25	$4.50
PTFE	2	$1.08	$2.16
	Total	$40.32	$43.59

Table 13: Bill of Materials for Mechanical Subsystem

<u>Electrical Subsystem</u>

The general outline of the electrical subsystem can be seen in the *Computer and Software Subsystem* section, where connections between electrical components and the Arduino Mega microcontroller are shown. Luckily, work necessary in order to ensure sound operation of selected electrical components is not reliant on heavy wiring or signal manipulation. Most of it comes down to ensuring that each component is set up, tested, and working properly, and that power consumption is properly accounted for. Necessary small-scale electrical component considerations are resistor and capacitor connections for use with LEDs as well as motors and driving boards. One additional decision not detailed in the aforementioned section is the sensor utilized for traffic and right of way detection. This decision is detailed below.

Traffic detection				
Criteria	**Weight**	**Ultrasonic Sensor**	**Camera**	**LiDAR**
Cost	2	+2	0	-2
Accuracy	5	+1	+1	+2
Complexity	3	0	-2	-2
Power Consumption	3	+1	0	-1
Ease of use	2	+2	0	-1
Total		+16	-1	-5

Table 14: Decision Matrix for Traffic Detection

The decision matrix in table 14 was used to evaluate three sensor options–ultrasonic sensor, camera and LiDAR–for traffic detection in the autonomous vehicle. Five criterias were considered: cost, accuracy, complexity, power consumption and ease of use, with accuracy having the highest weight due to its importance in ensuring reliable detection. Ultrasonic sensors were the most affordable at less than $10, followed by cameras and LiDAR. In terms of accuracy, LiDAR performed best while cameras and ultrasonic sensors followed closely. Ultrasonic sensors were the least complex to implement, consumed least power, and were the easiest to use, while camera and LiDAR required more effort. With a total of +16, ultrasonic sensors were chosen as the most suitable option. Cameras, scoring -1, provided high accuracy but were more expensive and complex, while LiDAR, scoring -5, was excluded due to its high cost and complexity.

To ensure obstacle avoidance and prevent collisions with other autonomous devices, the vehicle utilized an ultrasonic sensor. Ultrasonic sensors enable accurate, non-contact detection by emitting sound waves and measuring the time taken for the reflected wave to return. This capability allows them to reliably detect object distances and precise positioning, regardless of lighting or surface conditions. Additionally, ultrasonic sensors perform effectively in various environments, including dark and bright settings, and can detect a wide range of materials due to their ability to bounce off different surfaces. These sensors are also cost-effective and provide a reliable, affordable solution for obstacle detection and collision prevention [6].

In addition to use of an ultrasonic sensor for collision avoidance, the traffic detection capabilities was also aided and enhanced by the Pixy2 camera, a device whose inclusion will be detailed in the following paragraph. The Pixy2 camera was used for considerations where visual indicators are necessary in order to make traffic-related decisions. These include T-intersection direction-decisions, directional device recognition, and right-of-way.

As shown in the Computer/Software Subsystem, a decision matrix was used to evaluate different components for the vehicle's ability to detect the course line. Among the options considered, the PixyCam 2 was selected for its effectiveness in line following and line detection. The PixyCam 2 is a rotating and tilting camera designed to detect objects, lines, and intersections, offering many advantages in line-following applications. This device was chosen based on criteria such as connection, software compatibility, and power consumption. The PixyCam 2 is compatible with various microcontrollers and platforms, including Arduino.

After finalizing the motor selection and calculating the system's power needs, a decision matrix was developed to evaluate five battery options based on five weighted criteria: capacity, weight, cost, voltage suitability, and runtime. These criteria were chosen to reflect both the electrical and mechanical impacts of the battery on overall system performance, with voltage suitability and runtime being the most heavily weighted. The top two scoring options were the 12V 2600mAh Li-ion and the 12V 3000mAh Li-ion batteries, both achieving a total score of 22. These batteries provided ideal voltage compatibility for powering 12V motors and offered a good balance between weight, runtime, and cost. The 3000mAh version offered slightly longer runtime, while the 2600mAh version was slightly lighter. Lower-scoring options like the 7.4V 5000mAh LiPo were ruled out due to voltage being too low for the motors, despite offering the longest runtime. Similarly, the 9.6V 1100mAh LiFePO4 battery was lightweight but did not meet the minimum runtime required.

Based on this evaluation, the team selected the *12V 3000mAh Li-ion battery* as the final power source. It offers strong runtime, appropriate voltage for the motor driver and microcontroller, and is readily available for purchase. The decision matrix and criteria analysis used in this selection are shown in Table 15 below. This battery had supported vehicle operation for 6 continuous hours of testing, exceeding the two-hour minimum required of it.

Battery						
Criteria	**Weight**	**12V 2600mAh Li-ion**	**12V 3000mAh Li-ion**	**9.6V 3300mAh LiFePO4**	**9.6V 1100mAh LiFePO4**	**7.4V 5000mAh LiPo**
Capacity (mAh)	3	+1	+1	+1	+1	+2
Weight	4	0	0	0	+2	-2
Cost	2	+2	+2	0	0	-2
Voltage Suitability	5	+2	+2	+1	+1	0
Runtime	5	+1	+1	+1	-1	+2
Total		22	22	13	11	4

Table 15: Decision Matrix for Battery Selection

Initially, it was determined that the group would make use of a pre-fabricated PCB board in order to organize and connect all electrical components. After several design changes throughout the course of development, the finalized PCB design had to be discarded, and the team instead went with a terminal block shield designed specifically for the Arduino Mega board. This shield includes a soldered perfboard that houses connected components, notably stationary components, such as resistors and the motor driver board. The complete vehicle circuit can be seen in the figure below.

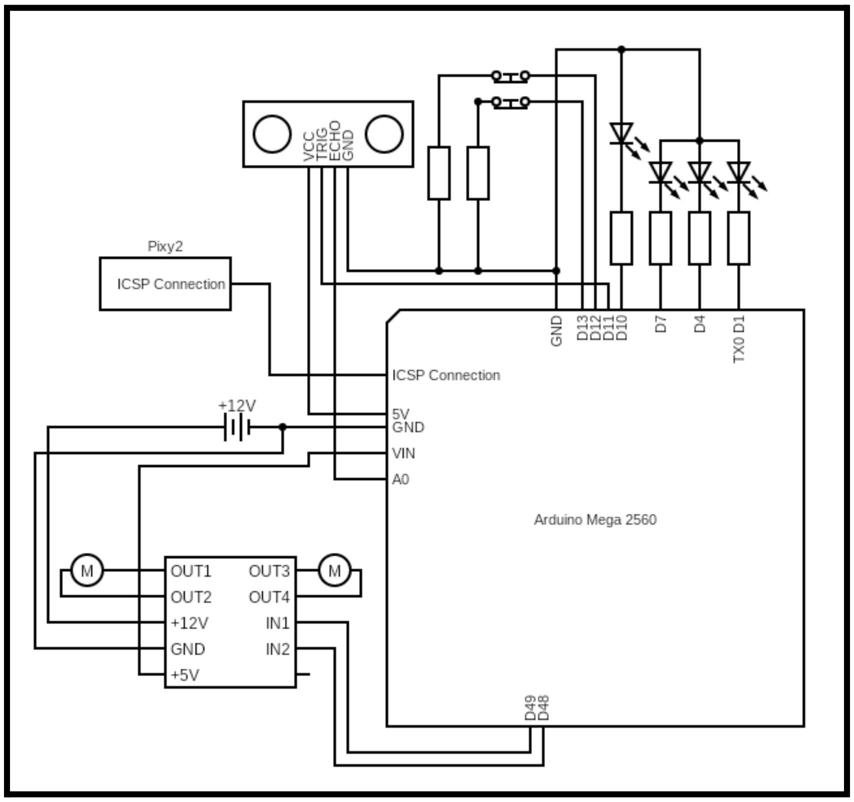


Figure 8. Electrical Circuit Schematic

As a $300 maximum development cost was targeted, the combined total of all necessary components was figured in order to meet this goal. Costs for all components of the electrical subsystem can be seen in the below table.

ECE Subsystem Estimated Bill of Materials		
Items	Quantity	Cost
Pixy2 Smart Vision Sensor	1	$69.90
Pixy2 Tilt Kit	1	$29.90
Ultrasonic Sensor	2	$6.00
LEDs	2	$0.10
Battery	1	$22
Arduino Mega	1	$23
24 AWG Wires	1	$9.61
Limit switch	2	$4.00
Terminal Block Shield	1	$18.00
Total		$182.51

Table 16: Estimated Bill of Materials for Electrical Subsystem

Computer/Software Subsystem

i. Overview:

The computer subsystem is the heart of the autonomous vehicle. Component wise, the vehicle consists of a Pixy2 camera, an HC-SR04 ultrasonic sensor, two LEDs (1 rgb, 1 red), and a L298N motor driver board. To connect those components, an Arduino Mega was selected. Using the Pixy2 in color-connected-components mode, the vehicle can detect signatures that correspond to the blue line along with red, yellow, and green LEDs. From the detected signatures the vehicle can determine and follow the blue line, stop at designated stop lines, and determine the correct turn direction at intersections.

ii. Initial Concepts & Theories:

The initial design for this subsystem revolved around the usage of the Pixy2's line-following mode. Using the most significant vector, the heading error was determined and the speeds of the motors were adjusted accordingly. The perk of using this mode was its innate ability to register intersections. Ideally, the Pixy2 would register a stop-line as an intersection. After stopping prior to it and determining which direction to go using color-connected-component mode, the next turn angle would be set to -90, 0, or 90. After the angle is set, the Pixy2 would resume line-following mode and complete the turn.

After approaching and stopping prior to the stop-line, the camera tilts up and determines the availability of a directional device by scanning for a yellow signature. If there is a yellow signature registered, then a green signature with a similar y-value is scanned for until it is found. The x-values for each signature (yellow & green) are then compared and the turn direction is determined. If the green LED is to the left of the yellow LED, then the vehicle turns left. If the green LED is to the right of the yellow LED, then the vehicle turns right.

In the case that a directional device is not detected at an intersection, it first checks to see if there are vehicles within the frame to determine the yield status. If there is a vehicle detected within the frame and it is to the left of ours, then we yield. If the vehicle is on our right, then we have the right of way and continue processing the intersection. In the case where we don't have to yield, the vehicle approaches the intersection and first checks to see if there is an available path forward. If there is an available path forward then it takes it. If there is not a forward path available, then the Pixy2 looks to the left and right, scanning the area of each direction and selecting the path with the smallest area as the correct way to go. If there is a vehicle to the left of ours and we have to yield, then the Pixy2 continuously scans for red signatures (the back of the vehicle) making sure it is either leaving the frame horizontally or getting smaller for a user-defined amount of frames. If the red signature completes those actions or disappears from the frame altogether, then the yield status is returned as clear and our vehicle is allowed to continue on.

The HC-S04 is utilized at the start of every loop, scanning for any obstacles in the front of the vehicle. Using the measured value obtained from the sensor, a scalar is determined that will adjust the speed of the motors given that the vehicle is following the line. If an obstacle is detected at a distance of 200mm then the vehicle will kill power to the motors. This is used to ensure that our vehicle completes the requirement of maintaining a 200mm gap between ours and another vehicle. If an object is detected within the 300-400mm range, then the scalar will decrease the PWM being applied to the motors until the minimum PWM value is reached.

iii. Design Choice Justification:

In order to come up with the initial theory, research had to be done to find the most optimal hardware and software components available to use in the project. To begin, the most important feature, line detection/following, had to be covered. To tackle this task, the best choice was to use a camera with a neural network. The table below showcases the trade-offs between the different options.

Line-Detection/Following Method								
Criteria	**Weight**	**IR Sensors**	**Camera with Neural Network**	**Webcam with Computer Vision**	**Reflect-ive Optical Sensors**	**Laser-Based Sensors**	**Capacitive / Inductive Sensors**	**Phototransistor w/ Blue Filters**
Cost	2	0	-4	-2	+2	-4	-4	0
Accuracy	5	0	+10	+10	-5	+10	+5	+5
Complexity	3	0	-3	-6	0	-3	-3	-3
Speed	2	0	+2	-2	0	-2	-2	0
		Total	+5	0	-3	+1	-4	+2

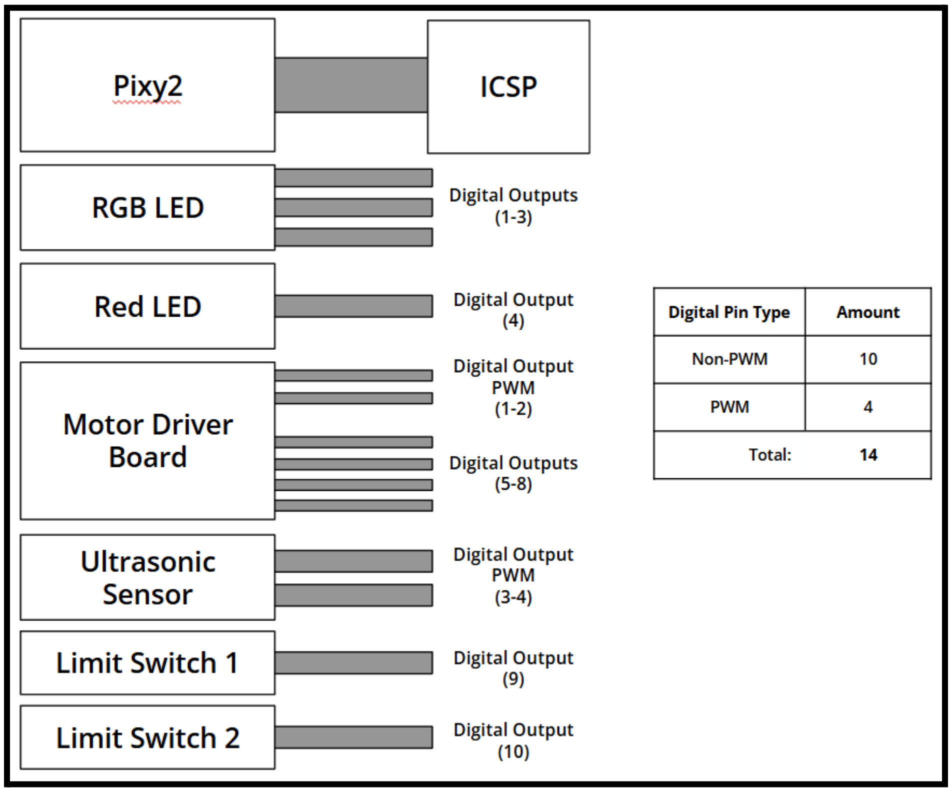
Table 17: Line-Detection/Following Method [11], [14], [15], [20], [25], [26], [29]

After the method of line detection was selected, another matrix was made to determine the specific component that was going to be utilized. In *Table 18*, three different cameras were evaluated and compared by their cost, software compatibility, power consumption, and connectivity. The result of the comparison was that the Pixy2 camera was the optimal choice consuming the least amount of power, while also being the easiest camera to connect to a microcontroller and program.

Line-Following Camera Selection				
Criteria	**Weight**	**HuskyLens**	**Pixy2**	**OpenMV Cam**
Cost	4	+4	-4	-8
Software Compatibility	1	+1	+2	+1
Power Consumption	3	-3	+6	+6
Connection	2	0	+4	+2
	Total	+2	+8	+1

Table 18: Line-Following Camera Selection

With a majority of the components being selected, an initial diagram was made to estimate the total amount of I/O pins that were needed. As can be seen in *Figure 8*, the estimated number of digital I/O pins was 13, with at least 4 of them being capable of PWM.



Digital Pin Type	Amount
Non-PWM	10
PWM	4
Total:	14

Figure 9: Estimated I/O Distribution

Using that as a reference, another matrix was created to compare different microcontrollers based on their cost, voltage range, amount of I/O ports, ease of use, and footprint. In *Table 5*, is the comparison for the following microcontrollers: Arduino UNO, Arduino Mega, Arduino Giga, Raspberry Pi Pico, ESP32, Micro:bit, and Bluepill. Out of all of those, the Arduino Mega achieved the highest rating with a total of +12. This was followed by the Arduino Giga and ESP32, where both achieved +11.

Microcontroller								
Criteria	**Weight**	**Arduino Uno**	**Arduino Mega**	**Arduino Giga**	**Raspberry Pi Pico**	**ESP32**	**Micro:bit**	**Blue Pill**
Cost	4	0	-4	-8	+8	+4	0	+4
Input Voltage Range	3	0	0	+3	-3	0	-6	-6
Digital I/O	5	0	+10	+10	+5	0	0	+5
Analog I/O	5	0	+10	+10	-10	+5	0	+5

Ease of Use	1	0	0	0	-2	-2	-2	-1
Footprint	2	0	-4	-4	+2	+4	0	+2
		Total	12	11	0	11	-8	9

Table 19: Decision Matrix for the Microcontroller [5-8], [22]

The Arduino Mega not only covers, but exceeds the I/O requirements that were originally estimated in Figure 8. Having a total of 54 digital I/O pins, 15 of which support PWM, will allow for the implementation of all current components and still have plenty of room for more if needed. With all of the components figured out, it was time to start working on the software aspect of this subsystem. For this autonomous car, the code was written within the Arduino IDE using C++. Also, important to note is that the components that went into the computer subsystem are accounted for in the electrical subsystem cost. The cost table showcasing the total price for this subsystem is in the prior section.

iv. Major Design Changes:

After running numerous tests, it became pretty clear that the line following mode of the Pixy2 was not going to be reliable for line following given the environment. Even manipulating the settings for the camera didn't aid the capabilities of the Pixy2 in any way. As a result of that, a decision was made to swap over from using line-following mode to color-connected-components. Using CCC mode has proved to be very challenging, but also very rewarding after fine-tuning. Below in table X, the justification for the decision to swap modes is made clear.

Using CCC mode to track and follow the line was a little more complicated than line-following mode. Using CCC allowed us to attach the different colors to signatures. For signatures 1-4, we taught the Pixy to recognize blue, yellow, green, and red respectively. The basis of line following is taking the centroid (center x,y values) of a blue line signature and determining the error between it and the center of the frame. After determining the error, it is sent through the PID, just like before, and returns a heading value that is used to adjust the speed of the motors.

Using DC motors instead of more precise ones did lead to some issues. Due to the different power bands for each of the motors, the speeds at different PWM values were obtained and from that an equation was extracted. The equation used to determine the ratio that relates the speed between the two motors is as follows:

$$Adjusted\ Speed\ Ratio\ =\ (3.76\epsilon - 8)\cdot corr^{3} + (-\ 2.07\epsilon - 5)\cdot corr^{2} + (4.57\epsilon - 3)\cdot corr + 0.608$$

Using this equation, the separate base speeds were removed and now we only use one base speed for both motors. The determined speed from calculations has to get multiplied by the ratio in order to obtain the correct PWM value for the left motor. By doing this, both motors now have the same power band and operate at the same speeds within the operable PWM range.

v. System Performance:

Overall the system can tackle each situation thrown at it when tested individually, but has problems when acting as a whole unit. The different situations that this system can handle include line following, stop line detection, intersection detection, intersection response, and yielding/right of way.

For the task of line following and stop line detection, the Pixy2 correctly identifies the line of blue tape and corrects the speed of the motors to ensure that the detected signature stays in the middle of the frame. Using determined thresholds, the camera can also accurately distinguish stop lines from the normal line and stop accordingly.

For intersection detection and response, the vehicle accurately determines the presence of a directional device by scanning for a yellow signature. If a directional device is detected, then the Pixy2 looks for a green signature with a similar y-value. Without a directional device, the vehicle first checks for the yield status before approaching the intersection and determining which route to go based on the registered area. By taking this approach, the vehicle can correctly navigate through any of the given intersections.

Yielding has not been practically tested but works as follows. The Pixy2 first scans for green signatures and then scans for red. Depending on the registered signatures, the pixy either yields if it has to, or continues movement in the case that it has right of way. In the case that the vehicle has to yield, the pixy continuously scans for a red signature and tracks it. Any change in height or positioning is monitored and if certain conditions are met then the yield is determined to be finished.

vi. Challenges Faced:

While going through and creating this computer subsystem, a great amount of challenges were encountered. This ranges from communication issues with the Pixy2 to broken servos and dissimilar motors. These problems required a great deal of thought to overcome.

After researching the communication issues, it was discovered that the Pixy2 pins that are connected to the ICSP port also connect to other digital pins that we were using for the limit switches. By utilizing these pins, that created a communication issue between the pixy and the board. After rearranging the pins, this problem was solved. Something to note is that there isn’t a lot of literature out there for people having similar issues.

While testing the vehicle as per usual, one day the ‘tilt’ servo motor stopped working. The cause of this is unknown and a new pan/tilt kit had to be ordered to remedy this problem. During the process of searching for solutions, it was discovered that a faulty servo motor connected to the Pixy2 could have also been a reason for the communication issue.

The issue that was had with the motors was discussed in a prior section. Basically, the power band, or RPM per PWM, was very different for each motor that was ordered. Two motors were ordered initially that had this problem, then later a third motor was purchased that had a difference in power band by a more than substantial amount. Using the ratio of motor speeds, it was possible to correct for the difference and have the original two motors function at near the same speeds.

Another major issue or problem that was had was signature teaching/recognition fine tuning using PixyMon. PixyMon V2 was not a user friendly experience and it took a lot of time to configure the settings to exactly what was wanted. After going through the hassle of fine tuning the parameters though, the camera performed up to par using CCC mode.

vii. Current Operation:

The final layout of the code is not too different from the original setup. Throughout the process of debugging, many features were included to increase the performance of the vehicle. Below in *figure 10*, a flowchart can be seen that shows the currently implemented main loop. Within that loop, the signature for the blue line is gathered and then it is determined if it is a normal line or a stop line. Depending on what is determined, the code branches off accordingly.

As can be seen on the right, if the detected signature was determined to be a stop line, then the code transitions to the "determineIntersection()" function. The purpose of this function is to scan for a yellow signature to determine if the current intersection contains a directional device or not. In *figure 11*, there is another flowchart that demonstrates the operation of this function.

As a result of the function, the code can take two routes. Either the intersection does or does not have a directional device. A flowchart for each possible route is showcased below, demonstrating the general idea of how each function operates.

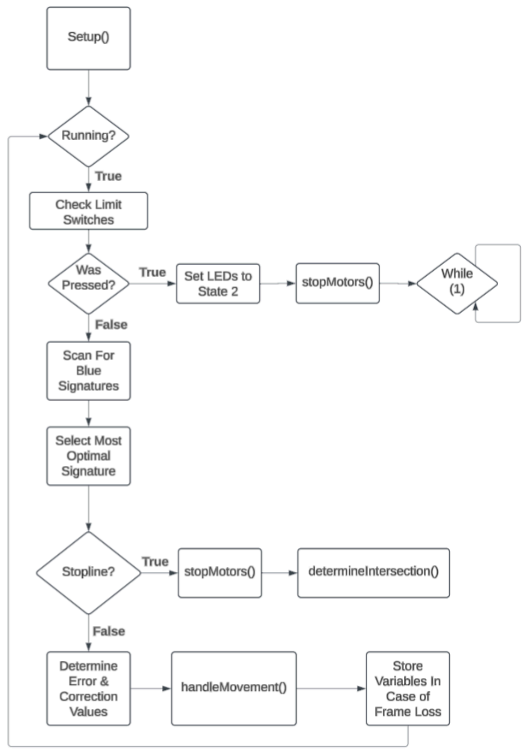


Figure 10: Main Loop Flowchart

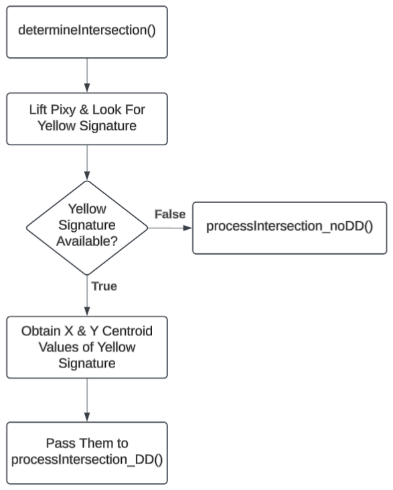

determineIntersection()
Lift Pixy & Look For Yellow Signature
Yellow Signature Available?
False
processIntersection_noDD()
True
Obtain X & Y Centroid Values of Yellow Signature
Pass Them to processIntersection_DD()

Figure 11: Intersection Type Determination

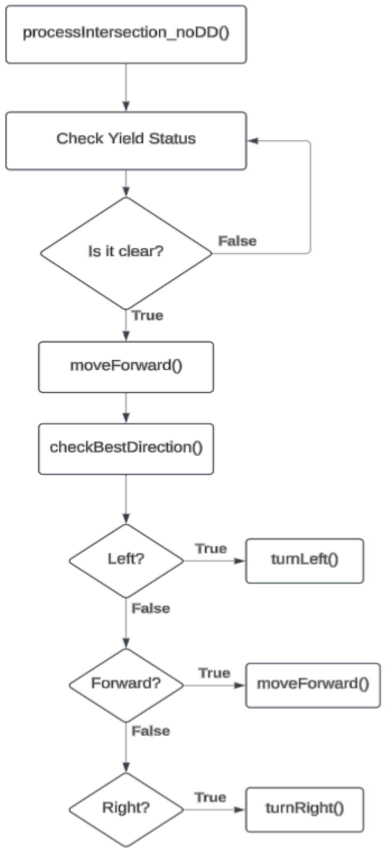


Figure 12: Intersection Type Determination

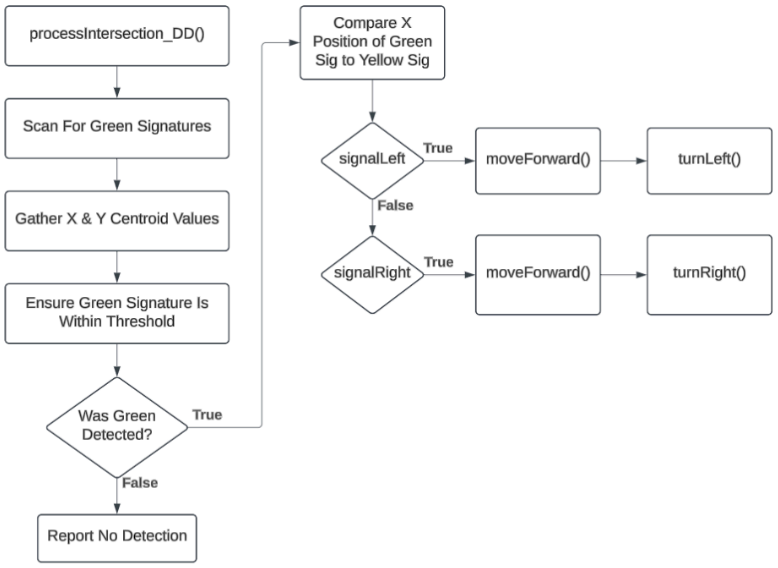


Figure 13: Intersection Type Determination

Another feature to note is the state machine that controls the LEDs on the vehicle. Within the setup function, the LEDs are initialized and set to S0, also known as the initial state. This state only toggles the rear LED on. At the beginning of the main loop the state is set to S1, known as the operating state, where the front RGB LED is set to green. If at any point the bumper switch is triggered, the LEDs enter into S2. This is the collision state of the vehicle and changes the front LED from green to red. A final state, S3, was added for debugging purposes and sets the front LED to blue. To the side, in *figure 14*, a visual representation of the LED state machine can be seen.

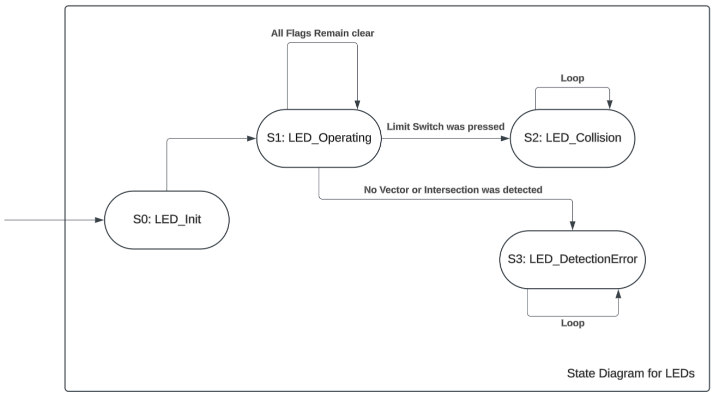


Figure 14: LED State Diagram

Discussion

a. Technical Discussion

The initial design addresses all requirements specified in the project description: a fully autonomous, line-following vehicle that recognizes and responds to external stimuli while generating stimuli of its own through external lighting.

To accommodate the line-following and external stimuli (such as other vehicles, turn singles, the blue painters tape, the intersections, etc) attribute of the autonomous vehicle, a camera and ultrasonic sensor were used to be able to detect these external stimuli with quick and reliable results. The components are also compact, lightweight, but in relation to the rest of this vehicle are some of the most expensive components to this vehicle.

b. Professional and Societal Context

i. Engineering Standard

Autonomous vehicles are subject to a range of engineering standards that ensure system reliability, safety, and accountability. For example, ISO 13849 focuses on functional safety in machinery and is often applied to emergency stop systems and protective components in automated platforms. UL 4600 provides a safety framework for fully autonomous systems by guiding validation of sensors, control logic, and fallback mechanisms. Additionally, SAE J3016 defines the levels of driving automation, from driver assistance to full autonomy, which helps classify the capabilities of different autonomous platforms. These standards shape how systems are architected, validated, and operated in the real world.

This project aligned with these standards where possible. The bumper switch and emergency motor shutoff align with ISO 13849 principles, while our use of line detection and stop-line response connects to the structured behaviors outlined in SAE J3016 for low-level automation. Although IPC-A-610 standards for PCB manufacturing weren't strictly followed due to our use of a shield board, proper soldering techniques and continuity checks were used to ensure safety and quality.

ii. Safety

Safety is a key requirement in autonomous vehicle development. Full-scale systems are expected to detect road features, avoid collisions, and respond correctly to unpredictable situations. This typically involves redundant sensing, fault tolerance, and extensive testing.

In this design, safety was addressed through both hardware and software. The front-mounted limit switch disabled motor power during a collision, and the ultrasonic sensors maintained a minimum following distance of 200 mm. These features support the same core objectives of collision avoidance and system deactivation found in larger autonomous platforms.

iii. Ethical Considerations

Ethical development of autonomous vehicles involves transparency, responsible use of resources, and environmental awareness. It also includes avoiding plagiarism, respecting software and hardware licensing, and documenting all sources and design decisions.

Throughout this project, all third-party components and example code were properly credited and used within license guidelines. Materials like ABS plastic were chosen for their reusability, and the design was structured to minimize waste. All logic and control algorithms were created by the team, with clear documentation to ensure accountability.

iv. Economic Factors

The adoption of additive manufacturing methods for the creation of this project lends itself to sustainability practices through minimizing extraneous material used. Though the chosen material (ABS) holds many benefits in terms of manufacturability and material properties, many issues exist in its recyclability and pollutive impact [18]. These impacts include chemical pollution, carbon-intensive production, human health effects, and the ever-growing issue of microplastic pollution [18]. Aside from the chassis, issues also exist in the lithium-ion battery technology chosen, though no clear and manufacturable solution has been found for this issue. Considering these impacts, a more sustainable plastic alternative may be in order if this project were to be produced on a larger scale.

Cost to implement the technology chosen would be considerable on a large scale, with the largest costs being associated with improval of system sophistication and testing of the vehicle before sale. Since a product such as a fully autonomous vehicle does not currently exist in the consumer market, impact on related jobs would be minimal, with more jobs being created than destroyed. As to the acceptance of the technology used, issues are predicted related to consumer trust.

v. Reliability

In the process of scaling this product for the mass market, it would be crucial to minimize failure of components and maximize reliability of the software subsystem. This necessitates consideration of the failures observed during development. These observed failures include failure of pan/tilt servo motors, quality inconsistencies in the chosen drive motors, wheel mount deterioration, 3D printing complications, and circuit connection issues. Taking these into account, further revision and changing of components may be necessary to ensure a reliable product. In addition, minimizing moving parts and consumer-side complexity would further improve reliability. Changes such as implementation of a power switch and exchanging perfboards for dedicated PCBs would assist in achieving this target.

On a larger scale, ensuring reliability of autonomous vehicle systems in all conditions is crucial before releasing to the mass market. Currently, issues exist pertaining to varying weather conditions and discernment of various obstacles. If a vehicle is to fail at any moment, the consequences may range from a minor annoyance to a total catastrophe.

vi. Aesthetics

The product detailed in this report has much opportunity to be sold to the mass market, having a clean and aesthetically pleasing design. The main exceptions to this design lay in the exposed electrical components visible from the outside of the vehicle. In addition, the majority of the electrical system is only separated by two covers on the top of the vehicle. While this is advantageous for serviceability, wire routing in the vehicle design is minimal, since the focus was chiefly functional. Another component of note is the pan/tilt kit, which was simply attached to the top of the vehicle. If scaling to a mass market, aesthetics would be improved by either creating a dedicated servo set for pan/tilt functionality or better concealing the provided kit. More permanent attachments and more sophisticated wire routing to minimize exposed wiring would be advantageous to the internal aesthetics of the vehicle.

vii. Potential Customers

This system was designed to be simple and accessible, which makes it a good fit for both educational and small-scale automation use. It could be useful for robotics teachers, STEM programs, or even hobbyists working on their own projects. The team chose parts that are affordable and easy to set up, so people with different experience levels could still work with it. Since things like camera-based tracking, ultrasonic sensors, and Arduino programming are already common in beginner robotics, this design could be a great starting point or template for others to build on. The team also tried to make the setup flexible enough to adapt to different needs or environments.

viii. Societal and Global Impact

Small-scale autonomous vehicles like this one have the potential to positively impact education, especially in introducing students to robotics, programming, and embedded systems. Due to its low cost and accessibility, the design could also be adapted for basic automation tasks in under-resourced settings or as a prototype for scalable applications. However, increased use of autonomous systems can contribute to concerns such as job displacement and electronic waste. These risks highlight the importance of responsible design practices, including the use of reusable materials and clear documentation to support long-term use and modification. Overall, this project emphasizes affordability, transparency, and adaptability, supporting broader goals of sustainable technology education and development.

c. <u>Information Literacy</u>

During the course of the project, the team obtained information from a wide variety of sources. These ranged from manufacturer data sheets and property tables to academic papers, online articles, and even other teams. Manufacturer data sheets provided detailed information on specifications, functions, and dimensions while property tables provided data on operating and/or material characteristics. In another vein, academic papers provided in-depth technical reports on similar projects and studies, while online articles gave plain-language accounts on using certain features, code functions, or techniques.

Despite the competitive nature of the project at hand, many teams still collaborated to exchange ideas and assist in troubleshooting complex issues. In this process, special care was taken to ensure that ideas and techniques were not directly taken without sufficient changes and/or credit to ensure originality. Through this collaborative nature with not only each other but also a rich mix of external sources, each group benefited from the sharing of knowledge; a common practice in both academia and the professional world.

d. Project Roles and Responsibilities

Project roles were determined by a number of factors, mainly, prior experience and areas of study. Our team had two mechanical engineering students, two electrical engineering students, and one computer engineering student and we decided that each member would work within the subsystems pertaining to these majors. Subsequently, the focuses within each subsystem were determined by experience. If one student was particularly comfortable with working in one area, they were to be assigned to this area. Little friction had resulted from the assignment of roles, and they were assigned quite easily, as each student believed their prior work had suited their respective roles. The final roles and responsibilities for the project have been distributed among group members, as seen in Table 16 below, to ensure the project's successful completion. Following this, a brief description of each project role is provided.

Group 8 Members	Project Role
Lucas Costello	CE Program Lead
James Haser (ME Lead)	ME CAD
Randa Oroo-Zro	EE PCB layout/Circuit Design & Hardware
Jon Tezak	ME Kinematics
Joseph Tituskin (ECE Lead)	EE Circuit Design

Table 20: Assigned Roles and Responsibilities

CE Program Lead: This role assumes the responsibility of developing computer programs that compile electrical subsystem functionalities into one cohesive system. Much of this work involves coding in a digital environment and machine training for necessary components.

ME CAD: This project role involves developing a number of digital models that represent the project. These models are used for planning, creating digital prototypes, and exporting for material printing. This work mostly pertains to the mechanical subsystem, outside of initial measurements for accommodating electrical peripherals and components. This role also encapsulates 3D printing responsibilities and tasks.

EE PCB layout/Circuit Design & Hardware: This role focuses on transforming circuit designs into physical implementations. Using schematics, this role involves creating and finalizing the PCB layout for the physical assembly of the design. Also, all processes pertaining to the wiring, testing, and usage of electrical components are assumed by this role.

ME Kinematics: This role encompasses the solution of motion-oriented design problems, especially when they pertain to structural components. These problems include mathematical ones pertaining to torque, acceleration, and turning, as well as those concerning the structure or chassis.

EE Circuit Design: This role assumes the responsibility of developing a circuit design necessary for the integration of the electrical components with the computer subsystem. All processes pertaining to the wiring, testing, and usage of electrical sensors and components are assumed by this role.

Conclusions and Recommendations

The final autonomous vehicle design successfully meets the outlined requirements of the Winter 2025 competition. The integration of mechanical, electrical and software subsystems enabled the vehicle to reliably follow a line, stop at intersections, and respond to directional devices. The use of the Pixy2 Camera and ultrasonic sensor allowed for precise environmental awareness, while the Arduino Mega controlled system behavior effectively.

Throughout the development process, several changes were necessary to address subsystem integration challenges. For instance, the original PCB design had to be replaced with a prototype shield board due to layout and timing constraints. Additionally, the initial plan to use the Pixy2's line following mode was changed to color-connected-components mode, which allowed for more accurate detection but required fine-tuning of the PID controller and error-handling logic.

Subsystem integration also introduced unexpected issues, such as motor inconsistencies and servo failures which were fixed by adjusting the code, changing hardware, and simplifying how things were mounted. Despite these challenges, the final prototype is stable and functional.

For future changes, a few improvements are recommended:

- Design and test the PCB layout earlier to allow time for revisions and integration.
- Use better wire management to clean up the layout and avoid loose connections.
- Find stronger or more reliable servo motors for the Pixy2 mount.
- Set up basic automated tests to check performance without relying only on manual testing.
- Transition more towards machine vision (OpenCV) instead of relying on the programs built into the Pixy2.

Overall, the project demonstrates a strong foundational approach to designing autonomous systems. With more time allocated to subsystem integration and refinement, future versions of the vehicle could achieve even higher performance and reliability.

Senior Design Project Expo and Competition

The project involved the design of a fully autonomous line-following vehicle developed for the Winter 2025 Senior Design competition. Although the vehicle did not qualify, testing demonstrated successful performance in several key areas. Stop line detection reached 100% when moving clockwise and 71% counter-clockwise. Vehicle speeds were measured at 227 mm/s without weights and 195 mm/s with weights, both within competition limits. Traffic operations showed partial success, with directional device recognition at 84%, T-intersection handling at 73%, and 4-way intersections at 51%. However, full-system integration challenges and unreliable obstacle detection from the ultrasonic sensor prevented consistent performance across full-course trials. If the project was revisited, alternate components would be selected to improve reliability and reduce integration complexity. However, the project was a valuable experience in debugging, team coordination, and system-level troubleshooting. If given more time, the team would have focused earlier on full-system integration and fallback behaviors to improve reliability under competition conditions.

References

[1] ISO, “Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design.” 2023. [Online]. Available: https://cdn.standards.iteh.ai/samples/73481/a2b27fd1dab8460fa3cef34426de7cce/ISO-13849-1-2023.pdf

[2] ISO, “Intelligent transport systems — Forward vehicle collision mitigation systems — Operation, performance, and verification requirements,” June. 2013. [Online]. Available: https://cdn.standards.iteh.ai/samples/45339/1a6b2e5bbf9f4393863e1b0d9ba0fe91/ISO-22839-2013.pdf

[3] SAE International, "SAE J3016 automated-driving graphic," May 15, 2020. [Online]. Available: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic

[4] NextPCB, “IPC-A-610: the standard for Acceptability of Electronic assemblies,” Dec. 30, 2024. [Online]. Available: https://www.nextpcb.com/blog/ipc-a-610

[5] “Autonomous Vehicle Technology | UL Standards & Engagement”, [Online]. Available: https://ulse.org/ul-standards-engagement/autonomous-vehicle-technology

[6] R. Ayala and T. Khan Mohd, “Sensors in Autonomous Vehicles: A Survey,” Journal of Autonomous Vehicles and Systems, vol. 1, no. 3, pp. 1–16, Nov. 2021, doi: https://doi.org/10.1115/1.4052991.

[7] 3dprinting.com, “What is 3D printing?,” [Online]. Available: https://3dprinting.com/what-is-3d-printing/. Accessed: Jan. 25, 2025.

[8] All3DP, J. O’Connell, “3D Printing Infill: The Basics for Perfect Results,” [Online]. Available: https://all3dp.com/2/infill-3d-printing-what-it-means-and-how-to-use-it/. Accessed: Jan. 25, 2025.

[9] 3DXTech Advanced Materials, “Technical Data Sheet: CarbonX™ CF-PLA 3D Printing Filament,” [Online]. Available: https://cdn.shopify.com/s/files/1/0625/4185/6821/files/CarbonX_CF_PLA_TDS_v3.pdf?v=1723942765. Accessed: Jan. 22, 2025.

[10] Aerospace Specification Metals, Inc., “Aluminum 6061-T6; 6061-T651,” asm.matweb.com, [Online]. Available: https://www.metalsdepot.com/aluminum-products/6061-aluminum-sheet-plate. Accessed: Jan. 20, 2025.

[11] Laminated Plastics, “Technical Data Sheet; Nylon,” [Online]. Available: https://www.secs.oakland.edu/~latcha/ME4999/PC-ABS%20-%20EN%20Data%20Sheet%20FDM%20Material.pdf. Accessed: Jan. 22, 2025.

[12] matweb, “Overview of materials for Nylon 66, 10% Carbon Fiber Filled,” [Online]. Available: https://www.matweb.com/search/datasheet_print.aspx?matguid=74fc8373189d42ebb0c35442433a4d16. Accessed: Jan. 22, 2025.

[13] matweb, “Overview of materials for Polylactic Acid (PLA) Biopolymer,” [Online]. Available: https://www.matweb.com/search/DataSheet.aspx?MatGUID=ab96a4c0655c4018a8785ac4031b9278&ckck=1. Accessed: Jan. 20, 2025.

[14] matweb, “Overview of materials for Polycarbonate/ABS Alloy, Unreinforced,” [Online]. Available: https://www.matweb.com/search/DataSheet.aspx?MatGUID=d16a64a389df424f8e53ed481e77477b&ckck=1. Accessed: Jan. 22, 2025.

[15] metalsdepot, “6061 Aluminum Sheet and Plate,” [Online]. Available: https://www.metalsdepot.com/aluminum-products/6061-aluminum-sheet-plate. Accessed: Jan. 20, 2025.

[16] C. Bhamra, “Comparative Study: Stepper Motors vs. Brushless DC Motors,” EZMotion, [Online]. Available: https://www.ezmotion.co/comparative-study-stepper-motors-vs-brushless-dc-motors. Accessed: Jan. 25, 2025.

[17] P. Millett, “Brushless Vs Brushed DC Motors: When and Why to Choose One Over the Other,” Monolithic Power Systems, [Online]. Available: https://www.monolithicpower.com/en/learning/resources/brushless-vs-brushed-dc-motors. Accessed: Jan. 25, 2025.

[18] N. Nyländen, “Recycling ABS plastic sustainably,” Sulapac [Online]. Available: https://www.sulapac.com/blog/replacing-abs-plastic-sustainably/. Accessed: Apr. 13, 2025.

[19] Arduino®, "Arduino® GIGA R1 WiFi Product Reference Manual," SKU: ABX00063, Jan. 2025. [Online]. Available: https://docs.arduino.cc/resources/datasheets/ABX00063-datasheet.pdf.

[20] Arduino®, "Arduino® Mega 2560 Rev3 Product Reference Manual," SKU: A000067, Jan. 2025. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf.

[21] Arduino®, "Arduino® Nano Product Reference Manual," SKU: A000005, Jan. 2025. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf

[22] Arduino®, "Arduino® UNO R3 Product Reference Manual," SKU: A000066, Jan. 2025. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

[23] Articulated Robotics, "Power Concepts," [Online]. Available: https://articulatedrobotics.xyz/tutorials/mobile-robot/hardware/power-theory/

[24] G. Sonal, P. Raninga, and H. Patel, "Design and implementation of RGB color line following robot," International Conference on Computing Methodologies and Communication, 2017.

[25] J. Sarwade, S. Shetty, A. Bhavsar, M. Mergu, and A. Talekar, "Line Following Robot Using Image Processing," Proceedings of the Third International Conference on Computing Methodologies and Communication, 2019.

[26] K. H. Ng, C. F. Yeong, E. L. M. Su, T. Y. Lim, and Y. Subramaniam, "Adaptive Phototransistor Sensor for Line Finding," International Symposium on Robotics and Intelligent Sensors, 2012.

[27] M. Z. A. Rashid, et al., "METAL LINE DETECTION: A NEW SENSORY SYSTEM FOR LINE FOLLOWING MOBILE ROBOT," Journal of Theoretical and Applied Information Technology, June. 2014.

[28] Micro:bit Educational Foundation, "BBC micro:bit v2 Datasheet," Version 1.2, Oct. 2020. [Online]. Available: https://raspberrypi.dk/wp-content/uploads/2020/10/BBC-microbit-v2-datasheet-v1.2.pdf

[29] Multicomp, "Standard LED, Red Emitting Colour," Part Number: MV5754A, Datasheet, Oct. 2011. [Online]. Available: https://www.farnell.com/datasheets/1498852.pdf

[30] Parallax, "TSL1401-DB (#28317): Linescan Camera Module" Datasheet, 2009. [Online]. Available: https://www.mouser.com/datasheet/2/321/28317-TSL1401-DB-Manual-369950.pdf?srsltid=AfmBOorzcRc6FzmXE2ZQCxgQ-aclcnxS83YlOMWPNXmr2v1cUhC5D0BI

[31] pixycam, "PIXY Documentation," [Online]. Available: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:start

[32] Raspberry Pi Ltd., "Raspberry Pi Pico Datasheet," Jan. 2021. [Online]. Available: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

[33] R. Alitappeh, A. Zabbah, M. Abdolmaleki, and K. Jeddisaravi, “Line Following Autonomous Driving Robot using Deep Learning,” 6th Iranian Conference on Signal Processing and Intelligent Systems, 2020.

[34] Shenzhen Normand Electronic Co., Ltd., "WS2815 Intelligent Control Integrated LED Light Source," Datasheet, Aug. 2018. [Online]. Available: http://www.normandled.com/upload/201808/WS2815%20LED%20Datasheet.pdf

[35] STMicroelectronics, "STM32F103x8 STM32F103xB: Medium-density performance line Arm®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces," Datasheet, DS5319 Rev 19, Sep. 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/cd00161566.pdf

[36] SuperLightingLED, "APA102 5050 Intelligent Control Integrated LED Light Source," Datasheet. [Online]. Available: https://www.superlightingled.com/PDF/APA102.pdf

[37] Worldsemi, "WS2812B Intelligent Control LED Integrated Light Source," Datasheet, 2012. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

Appendices

Phase 1 deliverables

Customer Needs and Functions

Needs / Functions	Priority	Notes
Footprint	3	The vehicle must have a compact design, with a maximum allowable footprint of 25 cm x 25 cm.
Reliability	10	High reliability is essential for autonomous functionality during the competition, ensuring the vehicle performs consistently across various conditions.
Affordable	9	The project must remain within the $300 budget constraint while maintaining the required functionality.
Line Tracing	10	The vehicle must accurately follow a 1-inch wide blue tape line.
Traffic Compatibility	10	Sensors must ensure the vehicle maintains a safe 200 mm distance from obstacles and yields appropriately at intersections.
Maneuverability	10	The vehicle must have excellent maneuverability to navigate tight corners, curves, and varying surface conditions.
Speed	2	minimal impact on performance.
Bumper Switch	5	Provides safety by detecting collisions.

Engineering Design Requirements and Validation

Category	Engineering Requirements	Importance	Target	Validation Method	Status
Structure	Footprint	High	< 25cm x 25cm	CAD analysis, Inspection	Complete
Structure	Motor Mount	Med	Withstand 15 Nmm	CAE Analysis	Complete
Motion	Turning Radius	High	0 < x < 250 mm	Estimate through analysis	Complete
Motion	Operating Speed	High	100-200 mm/s	Physical Validation	Complete
Line Tracing	Line Following Tolerance	High	Follow 1" wide painter's tape +/- 10 mm	Physical Validation	Complete
Line Tracing	Intersection and Turn Detection	Med	Stop 15 +/- 5 mm prior to stop line	Physical Validation	Complete
Traffic	Following Distance	Low	> 200 mm	Programming, Physical Validation	Complete
Traffic	Distance Sensor	High	> 300 mm	Component Datasheet	Complete
Traffic	Turn Signal Construction	High	< 25cm	CAD analysis, Inspection	Complete
Power	Input Power	Med	12 V	Microprocessor Power Ratings	Complete
Power	Operating life	High	> 2hr	Estimate through analysis	Complete
Cost	Design project cost	High	< $300	BOM	Complete

Project Task List

Task	Major
Decide on a complete component list	Electrical Engineering
Create flow charts	Electrical/Computer Engineering
Create a wiring diagram	Electrical Engineering
Verify all wiring/grounding	Electrical Engineering
Test all Subsystem	Electrical Engineering
Entire system flow chart operation	Computer Engineering
Pick a Programming Language	Computer Engineering
Coding	Computer Engineering
Complete Mechanical Design	Mechanical Engineering
Determine required tools/equipment	Mechanical Engineering

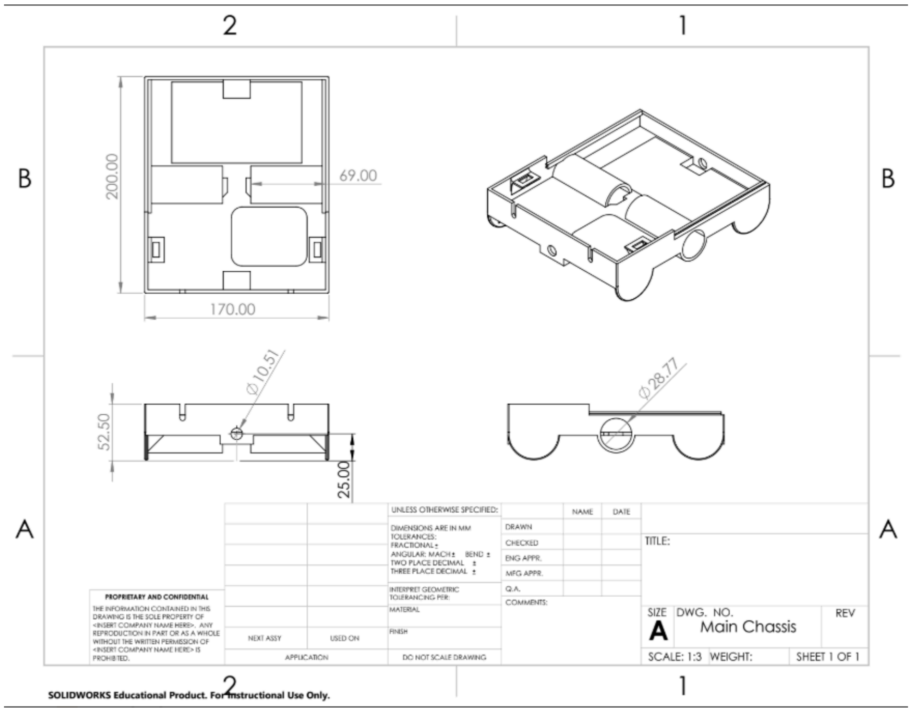
Preliminary Project Budget

Approved	Item	Estimated Cost	Actual Cost	Difference
Yes	Microcontroller	$28	$23	$5
Yes	Motors	$18	$30	$12
Yes	Motor Drivers	$20	$2.50	$17.50
Yes	Bumper Switch	$18	$12.24	$5.76
Yes	Ultrasonic Sensor	$10	$6	$4
Yes	Line Following Sensors (pixy)	$65	$70	$5
Yes	Blue Tape	$5	$5	$0
Yes	Mega Shield Board	$20	$18	$2
Yes	Chassis/Frame	$18	$22	$4
Yes	Tools	$10	$10	$0

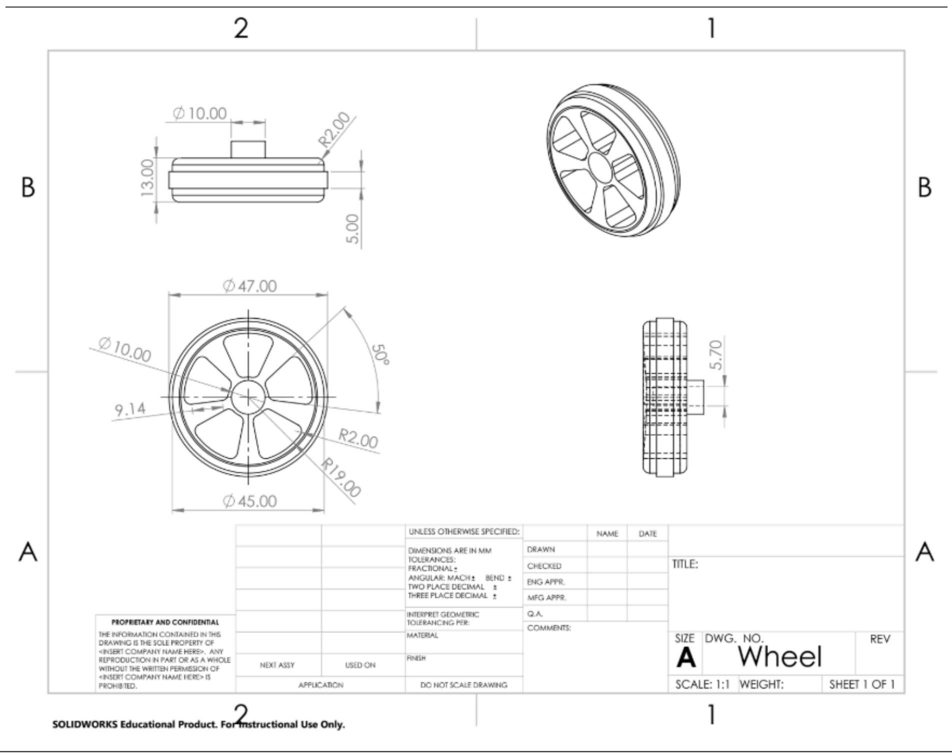
Phase 2 deliverables

Sketches and Schematics

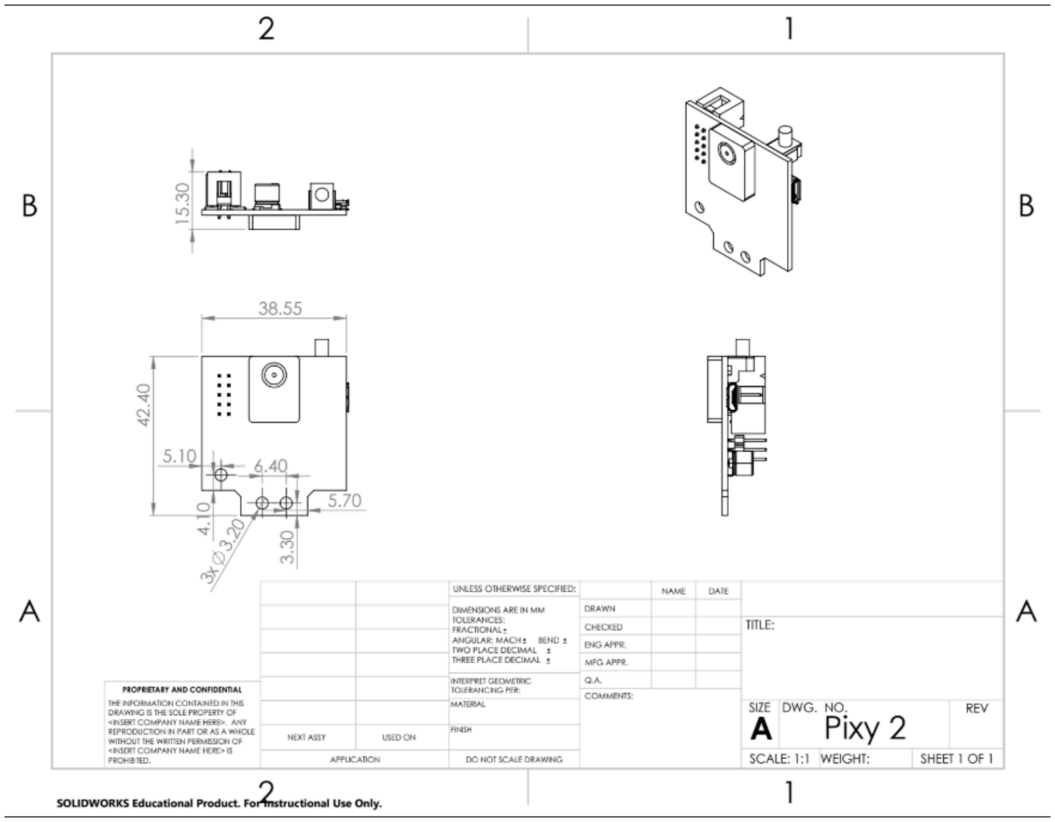
Drawings of Developed Components



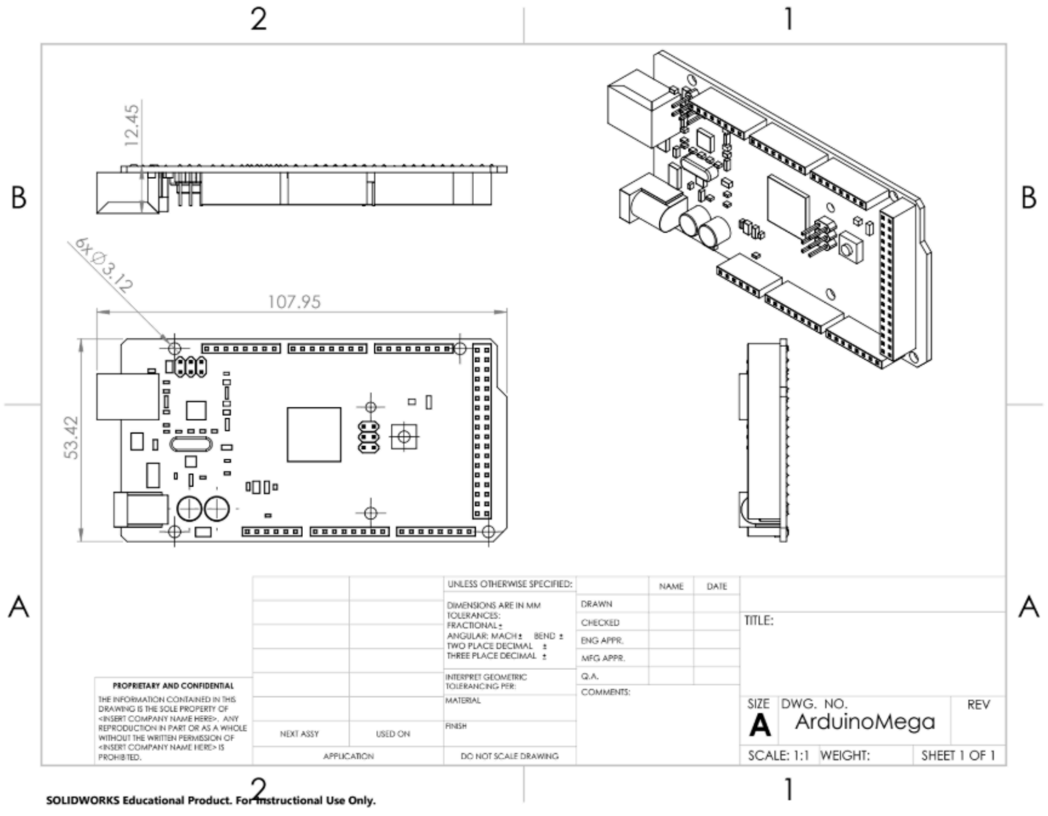
200.00
69.00
170.00
Ø10.51
52.50
25.00
Ø28.77
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
A
DWG. NO.
Main Chassis
REV
SCALE: 1:3
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.



2
1
B
A
⌀10.00
R2.00
13.00
5.00
⌀47.00
⌀10.00
50°
9.14
R2.00
R19.00
⌀45.00
5.70
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL±
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
A
Wheel
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.

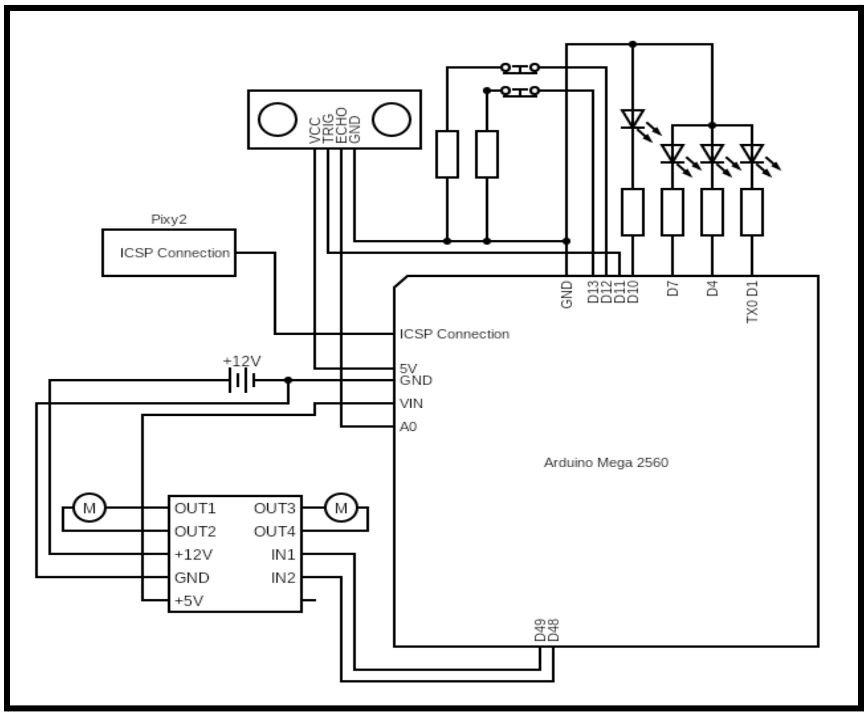


2
1
B
A
15.30
38.55
42.40
5.10
6.40
5.70
4.10
3X ⌀3.20
3.30
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL±
ANGULAR: MACH± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
A
Pixy 2
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.

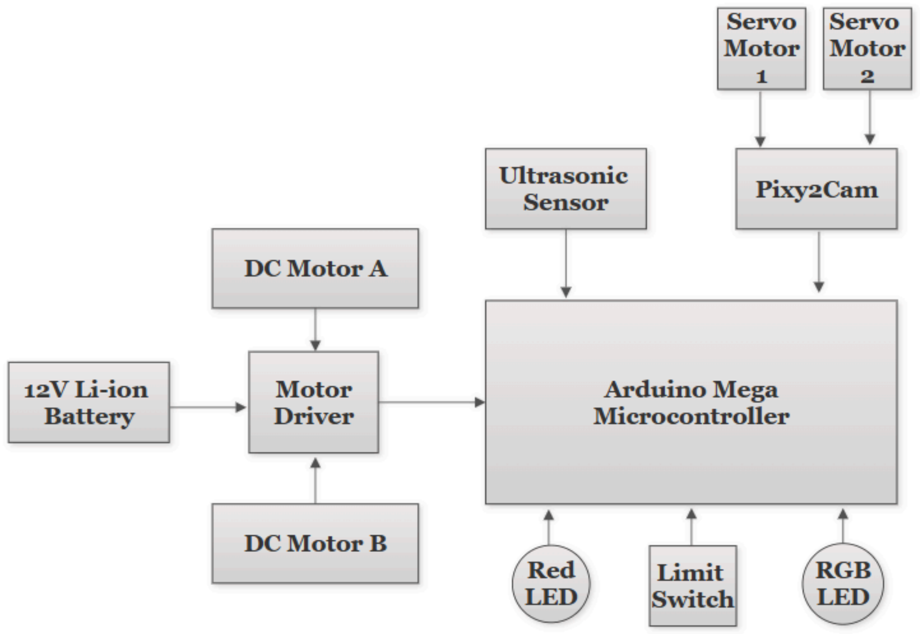


2
1
B
A
12.45
6x∅3.12
107.95
53.42
UNLESS OTHERWISE SPECIFIED:
NAME
DATE
DIMENSIONS ARE IN MM
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
A
ArduinoMega
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
SOLIDWORKS Educational Product. For Instructional Use Only.

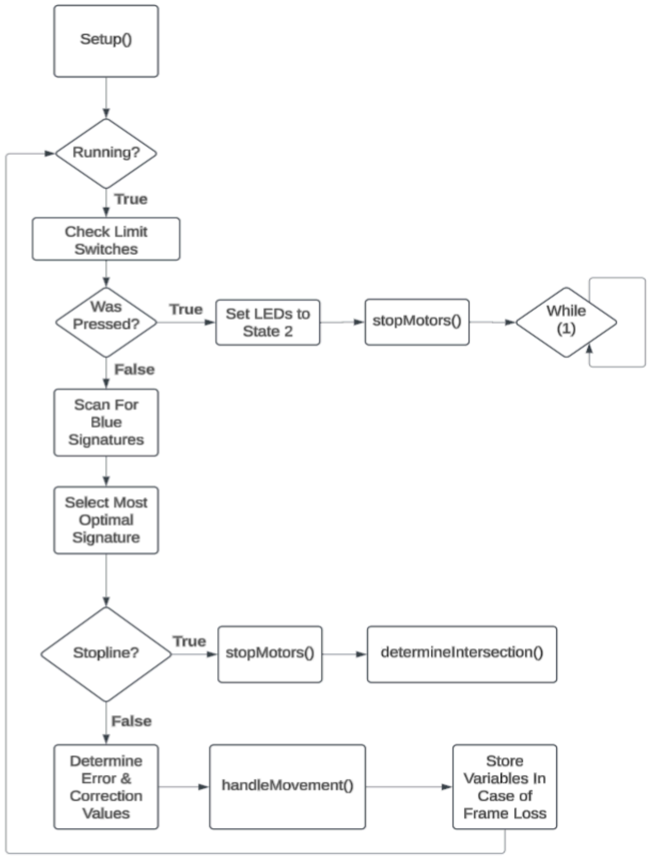
Electrical Circuit Schematic



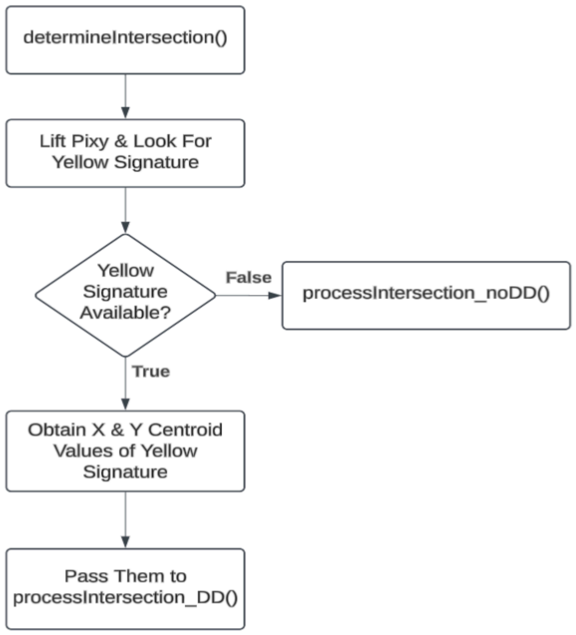
Block Diagram



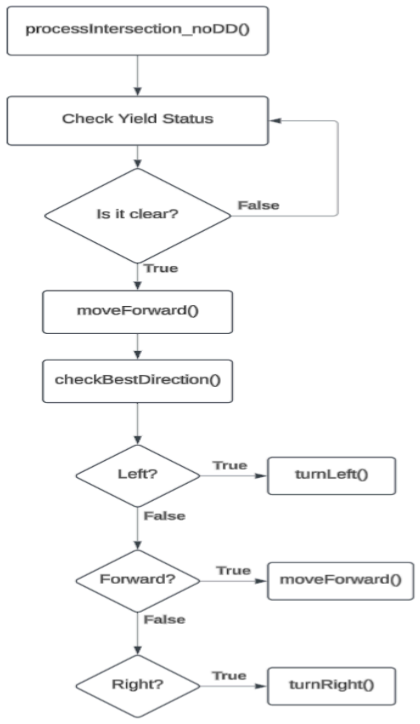
Flowcharts

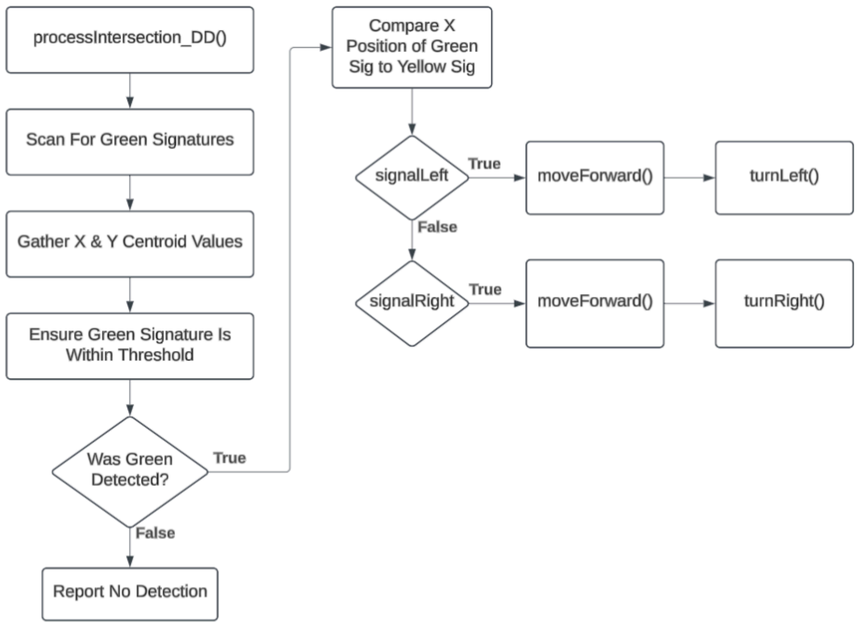


Setup()
Running?
True
Check Limit Switches
Was Pressed?
True
Set LEDs to State 2
stopMotors()
While (1)
False
Scan For Blue Signatures
Select Most Optimal Signature
Stopline?
True
stopMotors()
determineIntersection()
False
Determine Error & Correction Values
handleMovement()
Store Variables In Case of Frame Loss



determineIntersection()
Lift Pixy & Look For Yellow Signature
Yellow Signature Available?
False
processIntersection_noDD()
True
Obtain X & Y Centroid Values of Yellow Signature
Pass Them to processIntersection_DD()


processIntersection_noDD()
Check Yield Status
Is it clear?
False
True
moveForward()
checkBestDirection()
Left?
True
turnLeft()
False
Forward?
True
moveForward()
False
Right?
True
turnRight()


processIntersection_DD()
Scan For Green Signatures
Gather X & Y Centroid Values
Ensure Green Signature Is Within Threshold
Was Green Detected?
True
False
Report No Detection
Compare X Position of Green Sig to Yellow Sig
signalLeft
True
moveForward()
turnLeft()
False
signalRight
True
moveForward()
turnRight()

Decision Matrices

Material Manufacturing Method					
Criteria	**Weight**	**3D Printing**	**Waterjet Plastic Plates**	**CNC Plastic Plates**	**Laser Cut Plates**
Cost	3	+6	+3	+3	+3
Weight	5	+10	+5	+5	+5
Dimensional Stability	1	+2	+1	+2	+2
Eco-Friendly / Waste	2	-2	-4	-4	-4
	Total	+16	+5	+6	+6

Material Choice								
Criteria	**Weight**	**Alumi-num**	**PLA**	**ABS Plastic**	**Carbon Fiber PLA**	**PC-ABS**	**Nylon 6**	**Nylon Carbon Fiber**
Cost	4	0	-4	-4	-4	-8	-8	-8
Weight	3	0	+0	+6	+3	+3	+6	+3
Durability	2	0	-4	-2	-2	-2	-2	-2
		Total	-8	+0	-3	-7	-4	-7

Locomotion Type				
Criteria	**Weight**	**Wheels**	**Tracks**	**Legs**
Cost	4	1	-2	-1
Weight	5	1	1	1
Footprint	3	2	-2	2
Complexity	1	2	-2	-2
Speed	2	2	1	1
	Total	21	-9	7

Quantity of Best Locomotion Type				
Criteria	**Weight**	**2x**	**3x**	**4x**
Cost	4	1	0	-1
Weight	5	1	0	-1
Footprint	3	1	0	-1
Complexity	1	2	1	2
	Total	14	1	-10

Support Mechanism				
Criteria	**Weight**	**Felt Tape**	**Ball Bearing**	**Micro Controlled**
Cost	4	8	4	-4
Weight	5	10	5	10
Footprint	3	3	3	6
Complexity	1	2	1	-2
	Total	23	13	10

Support Mechanism (revised)				
Criteria	**Weight**	**Felt Tape**	**PTFE**	**Ball Bearing**
Cost	1	0	-1	-2
Weight	3	0	-3	-6
Footprint	2	0	4	4
Friction	3	0	6	3
Complexity	2	0	-2	-4
	Total	0	4	-5

Wheel/Floor Interaction Method				
Criteria	**Weight**	**Raw ABS**	**Electrical Tape**	**Rubber Tire**
Cost	1	2	1	-2
Traction	3	-6	3	3
Complexity	2	4	2	-2
Durability	2	2	-2	2
	Total	-2	2	1

Wheel/Floor Interaction Method (Revised)			
Criteria	**Weight**	**Electrical Tape**	**Rubber Tire**
Cost	1	0	-1
Traction	3	0	6
Complexity	2	0	-2
Durability	2	0	4
	Total	0	7

Electric Motor Type				
Criteria	**Weight**	**DC Brushed**	**DC Brushless**	**Stepper Motor**
Cost	3	3	-3	3
Power	4	4	-4	0
Efficiency	2	-2	2	-2
Operating Speed	1	0	0	-1
	Total	5	-5	0

Motor Design Change			
Criteria	**Weight**	**Original Motor**	**Motor with Gearbox**
Cost	2	0	-4
Torque	3	0	3
Efficiency	1	0	-1
Operating Speed	4	0	8
	Total	0	6

Traffic detection				
Criteria	**Weight**	**Ultrasonic Sensor**	**Camera**	**LiDAR**
Cost	2	+2	0	-2
Accuracy	5	+1	+1	+2
Complexity	3	0	-2	-2
Power Consumption	3	+1	0	-1
Ease of use	2	+2	0	-1
Total		+16	-1	-5

Battery						
Criteria	**Weight**	**12V 2600mAh Li-ion**	**12V 3000mAh Li-ion**	**9.6V 3300mAh LiFePO4**	**9.6V 1100mAh LiFePO4**	**7.4V 5000mAh LiPo**
Capacity (mAh)	3	+1	+1	+1	+1	+2
Weight	4	0	0	0	+2	-2
Cost	2	+2	+2	0	0	-2
Voltage Suitability	5	+2	+2	+1	+1	0
Runtime	5	+1	+1	+1	-1	+2
Total		22	22	13	11	4

Line-Detection/Following Method								
Criteria	**Weight**	**IR Sensors**	**Camera with Neural Network**	**Webcam with Computer Vision**	**Reflect-ive Optical Sensors**	**Laser-Based Sensors**	**Capacitive / Inductive Sensors**	**Phototransistor w/ Blue Filters**
Cost	2	0	-4	-2	+2	-4	-4	0
Accuracy	5	0	+10	+10	-5	+10	+5	+5
Complexity	3	0	-3	-6	0	-3	-3	-3
Speed	2	0	+2	-2	0	-2	-2	0
Total			+5	0	-3	+1	-4	+2

Line-Following Camera Selection				
Criteria	**Weight**	**HuskyLens**	**Pixy2**	**OpenMV Cam**
Cost	4	+4	-4	-8
Software Compatibility	1	+1	+2	+1
Power Consumption	3	-3	+6	+6
Connection	2	0	+4	+2
Total		+2	+8	+1

Microcontroller								
Criteria	**Weight**	**Arduino Uno**	**Arduino Mega**	**Arduino Giga**	**Raspberry Pi Pico**	**ESP32**	**Micro:bit**	**Blue Pill**
Cost	4	0	-4	-8	+8	+4	0	+4
Input Voltage Range	3	0	0	+3	-3	0	-6	-6
Digital I/O	5	0	+10	+10	+5	0	0	+5
Analog I/O	5	0	+10	+10	-10	+5	0	+5
Ease of Use	1	0	0	0	-2	-2	-2	-1
Footprint	2	0	-4	-4	+2	+4	0	+2
Total			12	11	0	11	-8	9

Bills of Material for each subsystem and the complete system

ECE Subsystem Estimated Bill of Materials		
Items	Quantity	Cost
Pixy2 Smart Vision Sensor	1	$68.62
Pixy2 Tilt Kit	1	$29.95
Ultrasonic Sensor	2	$6.00
LEDs	2	$0.10
Battery	1	$22
Arduino Mega	1	$23
Limit switch	2	$12.24
Terminal Block Shield	1	$18.00
Motor Control Board	1	$2.50
Total		$183.41

ME Subsystem Bill of Materials			
Items	Quantity	Cost	Ext. Cost
ABS Filament	0.315 kg (w/ 20% infill)	$21.99/kg	$6.93
Electric Motor	2	$15	$30
Bumper Spring	2	$2.25	$4.50
PTFE	2	$1.08	$2.16
	Total	$40.32	$43.59

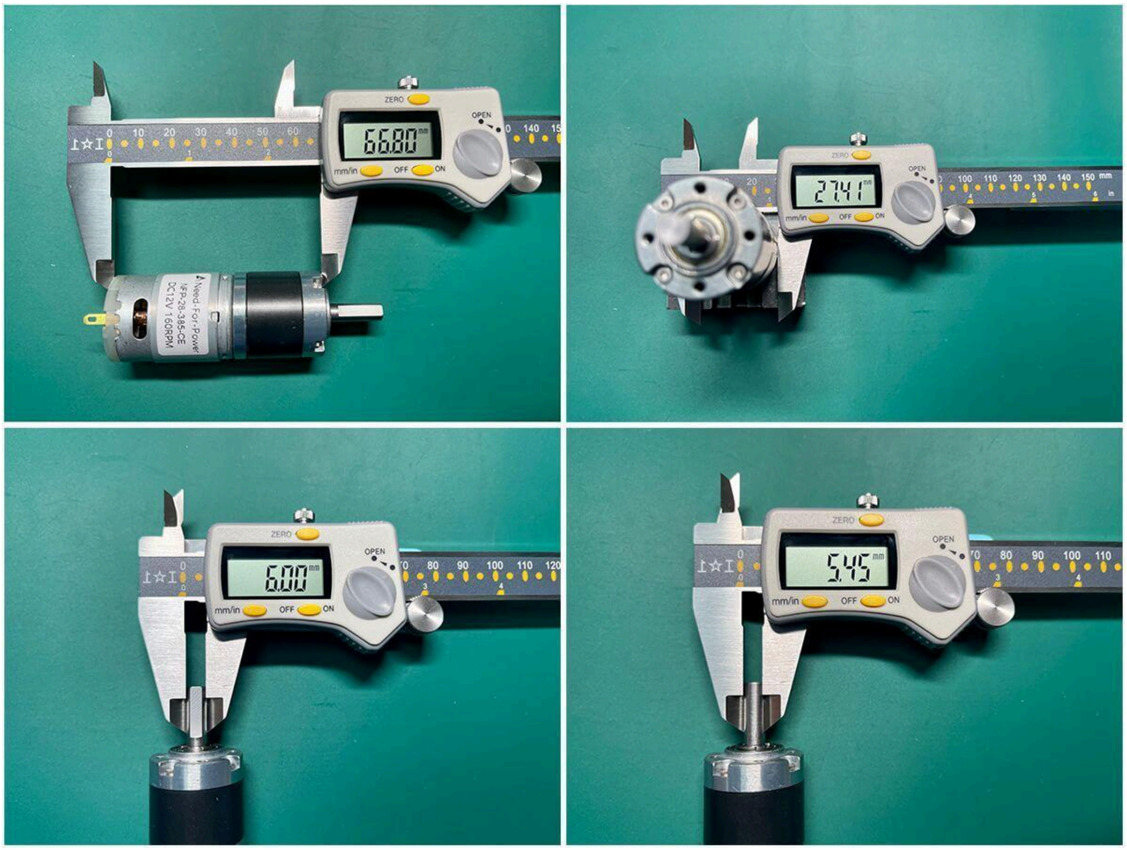
Complete System Bill of Materials			
Materials	Quantity	Cost	Ext. Cost
ABS Filament	1	$21.99	$21.99
PTFE Standoff	2	$1.08	$2.16
Limit Switch	2	$6.12	$12.24
Pixy2 Camera	1	$68.62	$68.62
Pixy2 Pan/Tilt Kit	1	$29.95	$29.95
Ultrasonic Sensor	1	$5.95	$5.95
Battery	1	$22.00	$22.00
Perfboard	1	$6.00	$6.00
Electric Motor	2	$15.00	$30.00
RGB LED	2	$1.15	$2.30
Motor Control Board	1	$2.50	$2.50
Arduino Mega	1	$23.00	$23.00
Riblok Clips	2	$0.03	$0.06
Dual Lock Velcro	1	$2.00	$2.00
Plexiglass Bumper	1	$0.40	$0.40
Tire	2	$1.00	$1.00
Total:			$229.17

Motor Datasheet

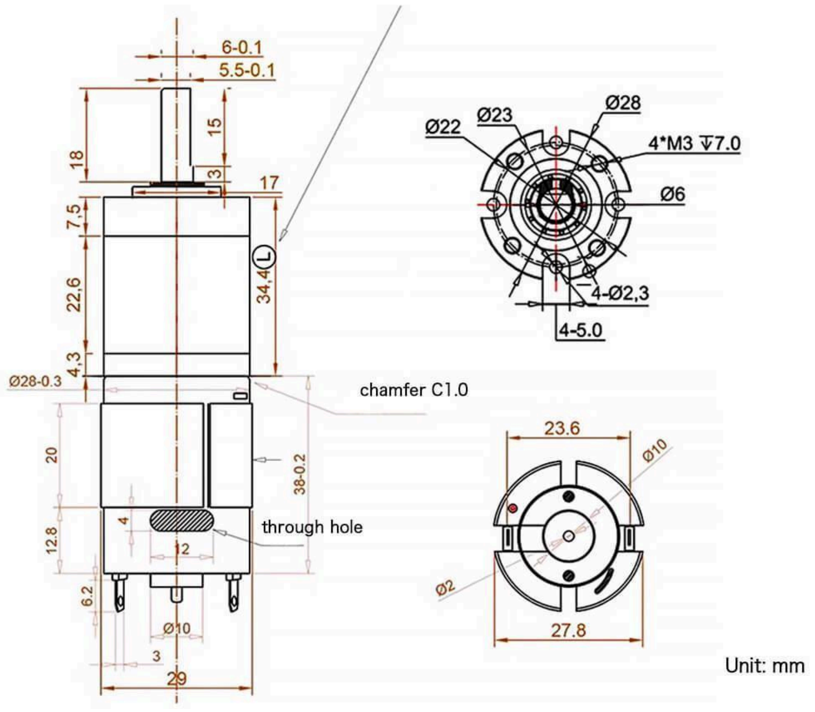
Gear Reduction Motor NFP-GA28Y-385 Specifications

Motor Model	NFP-GA28Y-385
Rotation	CW & CCW
Rated voltage	12V DC / 24V DC
No-load speed	10rpm – 2,000rpm
Out power	6W / 12W
Max torque	35Kg.cm
No-load current	0.25A / 0.5A
Output shaft diameter	6mm
Weight	185g – 210g

Ratio	No Load		Rated Load			Stall	
i : 1	Speed	Current	Speed	Current	Torque	Torque	Current
4	1000 rpm	0.05A	725 rpm	0.25A	0.3kg.cm	0.7kg.cm	0.8A
16	250 rpm	0.05A	180 rpm	0.25A	0.8kg.cm	2.5kg.cm	0.8A
25	160 rpm	0.05A	120 rpm	0.25A	1.5kg.cm	4kg.cm	0.8A
64	62 rpm	0.05A	45 rpm	0.25A	3kg.cm	9kg.cm	0.8A
88	45 rpm	0.05A	33 rpm	0.25A	4kg.cm	12kg.cm	0.8A
138	29 rpm	0.05A	21 rpm	0.25A	6kg.cm	20kg.cm	0.8A
400	10 rpm	0.05A	7 rpm	0.25A	20kg.cm	56kg.cm	0.8A



Ratio	4:1	16:1	25:1	64:1	88:1	138:1	400:1
L(mm)	23	28.5		34.4			40



Screw/Terminal Block Shield Board for Arduino Mega

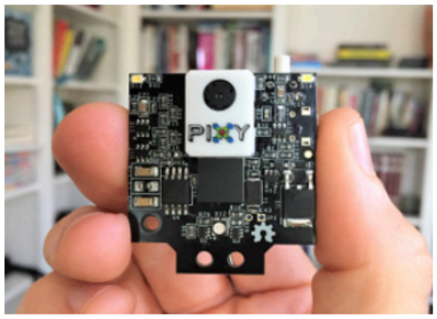
Product Description

Prototype Screw/Terminal Block Shield Board Kit For Arduino MEGA-2560 R3.

The Kit include Prototype PCB, Terminal Blocks, Female Header Sockets.
Unassembled, variety mount methods, you can make flexible use.
PCB:
FR-4 fiber glass PCB, dual copper layers, size 120mm x 78mm.
Pads grid / pitch 2.54mm / 0.1", hole diameter 0.8mm.
Terminal block pitch 3.5mm / 0.138", hole diameter 1.2mm.
Terminal Block:
Pitch 3.5mm / 0.138" mini terminal blocks, total 84 pins(3pole x24pcs, 2pole x6pcs).
Wire range 26~16AWG, strip length 5mm, screws M2 steel, pin header and cage brass.
Female Header Socket:
Pitch 0.1"/2.54mm, pin length 15mm / 0.59"
Single row 10pin x1pcs, singal row 8pin x 5pcs, dual row 18pin x1pcs.

Pixy2Cam

Pixy2 Overview



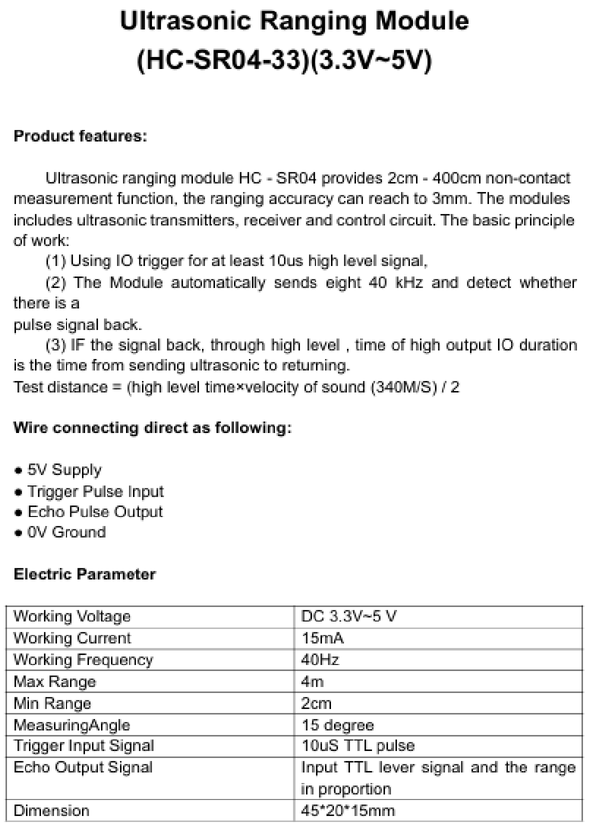
Pixy2 is the second version of Pixy. It's faster, smaller and more capable than the original Pixy, adding line tracking/following algorithms as well as other features. Here's what we've added to Pixy2:

- Pixy2 detects lines, intersections and small barcodes, intended for line-following robots
- Improved framerate – 60 frames-per-second
- Tracking algorithms have been added to color-based object detection
- Improved and simplified libraries for Arduino, LEGO Mindstorms EV3, Raspberry Pi and other controllers
- Integrated light source

And of course, Pixy2 does everything that the original Pixy can do:

- Small, fast, easy-to-use, low-cost, readily-available vision system
- Learns to detect objects that you teach it
- Connects to Arduino with included cable. Also works with LEGO Mindstorms EV3, Raspberry Pi, BeagleBone and similar controllers
- All libraries for Arduino, LEGO Mindstorms EV3, Raspberry Pi, etc. are provided
- C/C++ and Python are supported
- Communicates via one of several interfaces: SPI, I2C, UART, USB or analog/digital output
- Configuration utility runs on Windows, MacOS and Linux
- All software/firmware is open-source GNU-licensed
- All hardware documentation including schematics, bill of materials, PCB layout, etc. are provided

Ultrasonic Sensor Datasheet



Ultrasonic Ranging Module (HC-SR04-33)(3.3V~5V)

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a
pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340M/S) / 2

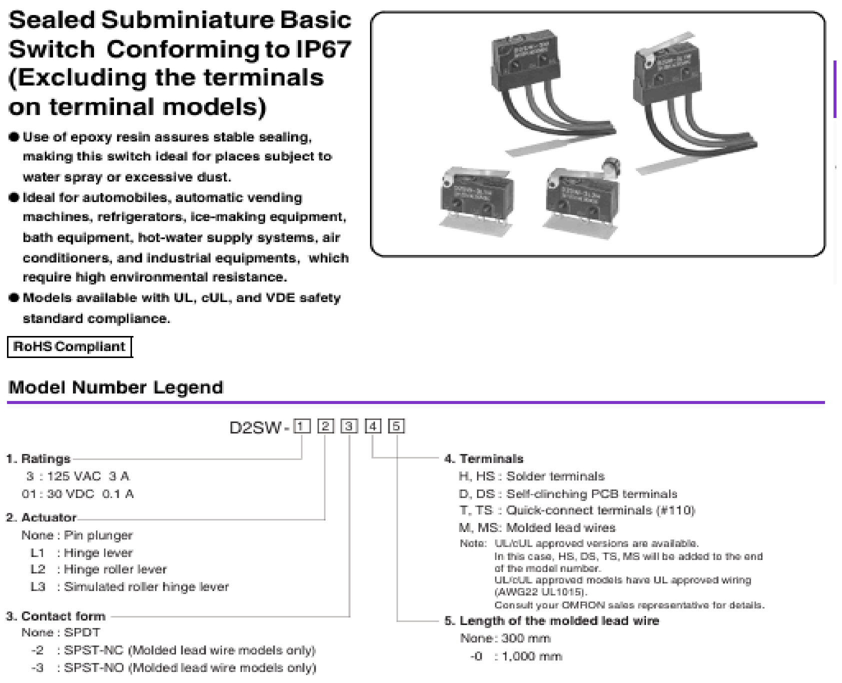
Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

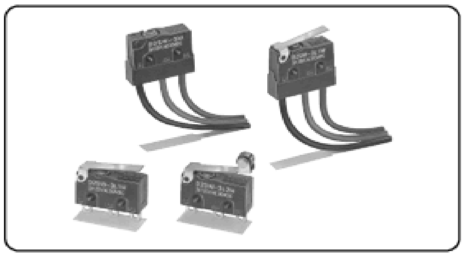
Electric Parameter

Working Voltage	DC 3.3V~5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Limit Switch Datasheet



Sealed Subminiature Basic Switch Conforming to IP67 (Excluding the terminals on terminal models)



- **Use of epoxy resin assures stable sealing, making this switch ideal for places subject to water spray or excessive dust.**
- **Ideal for automobiles, automatic vending machines, refrigerators, ice-making equipment, bath equipment, hot-water supply systems, air conditioners, and industrial equipments, which require high environmental resistance.**
- **Models available with UL, cUL, and VDE safety standard compliance.**

RoHS Compliant

Model Number Legend

D2SW-$\boxed{1}\ \boxed{2}\ \boxed{3}\ \boxed{4}\ \boxed{5}$

1. Ratings
3 : 125 VAC 3 A
01 : 30 VDC 0.1 A

2. Actuator
None : Pin plunger
L1 : Hinge lever
L2 : Hinge roller lever
L3 : Simulated roller hinge lever

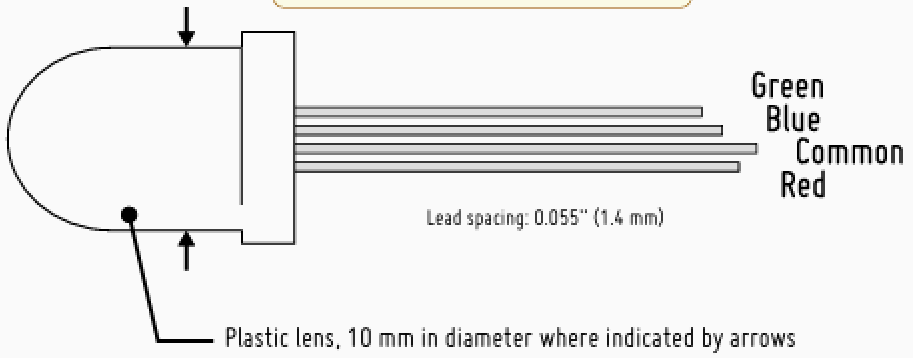
3. Contact form
None : SPDT
-2 : SPST-NC (Molded lead wire models only)
-3 : SPST-NO (Molded lead wire models only)

4. Terminals
H, HS : Solder terminals
D, DS : Self-clinching PCB terminals
T, TS : Quick-connect terminals (#110)
M, MS: Molded lead wires
Note: UL/cUL approved versions are available.
In this case, HS, DS, TS, MS will be added to the end of the model number.
UL/cUL approved models have UL approved wiring (AWG22 UL1015).
Consult your OMRON sales representative for details.

5. Length of the molded lead wire
None: 300 mm
-0 : 1,000 mm

10 mm Ultrabright RGB LEDs with Diffused Lenses

Evil Mad Science LLC
evilmadscience.com

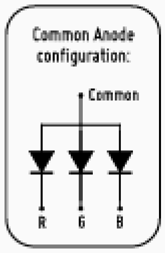


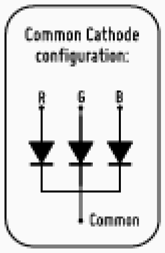
Plastic lens, 10 mm in diameter where indicated by arrows

- Milky-white diffused lens gives wide viewing angle and excellent color mixing. Best lens choice for indoor viewing.

LED Element	Wavelength (Peak, nm)	Viewing Angle	Forward Current (mA)		Forward Voltage (V_F)		Reverse Voltage (V_R)	Luminous Intensity (mCd)	
			Cont.	Peak*	Typical	Max		Min	Typical
Red	630	30°	30	150	2.1	2.5	5	200	500
Green	525	30°	30	150	3.1	3.8	5	300	1000
Blue	430	30°	30	100	3	4.2	5	300	800

Available in either common-anode or common-cathode configuration.





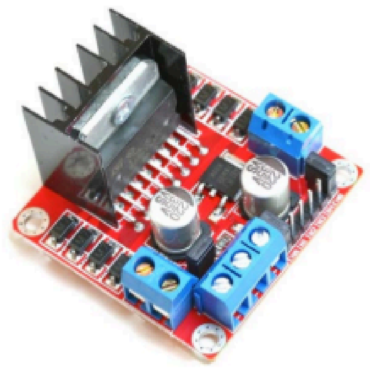
Manufacturer: Betlux Electronics www.betlux.com
Manf. Part No: BL-L105RGBW-CA (common anode)
Manf. Part No: BL-L105RGBW-CC (common cathode)

* Peak current is with limited duty cycle: 100 µs @ 1 kHz

** Note that luminous intensity accounts for both human eye response and viewing angle

Updated July 19, 2011

L298N Motor Driver Datasheet



SKU: MDU-1049

Brief Data:

- Input Voltage: 3.2V~40Vdc.
- Driver: L298N Dual H Bridge DC Motor Driver
- Power Supply: DC 5 V - 35 V
- Peak current: 2 Amp
- Operating current range: 0 ~ 36mA
- Control signal input voltage range :
- Low: -0.3V ≤ Vin ≤ 1.5V.
- High: 2.3V ≤ Vin ≤ Vss.
- Enable signal input voltage range :
    - Low: -0.3 ≤ Vin ≤ 1.5V (control signal is invalid).
    - High: 2.3V ≤ Vin ≤ Vss (control signal active).
- Maximum power consumption: 20W (when the temperature T = 75 ℃).
- Storage temperature: -25 ℃ ~ +130 ℃.
- On-board +5V regulated Output supply (supply to controller board i.e. Arduino).
- Size: 3.4cm x 4.3cm x 2.7cm

Code Listings

```
#include <NewPing.h>
#include <PIDLoop.h>
#include <Wire.h>
#include <Pixy2.h>

// Pin Declarations
  // Limit Switch
    const int limit_switch_1  = 44;
    const int limit_switch_2    = 45;
  // Ultrasonic Sensor
    const int us_trig_pin   = 5;
    const int us_echo_pin   = 6;
  // Motor Pins
    const int LeftMotor_PWM   = 10;
    const int RightMotor_PWM  = 9;
    const int LeftMotor_EN    = 35;
    const int RightMotor_EN   = 34;
  // LED Pins
    const int rearLed   = 26;
    const int rgbRed    = 27;
    const int rgbGreen  = 28;
    const int rgbBlue   = 29;

// Other Variables
  // Pixy2 Signatures
    const uint16_t BLUE   = 1;
    const uint16_t YELLOW = 2;
    const uint16_t GREEN  = 3;
    const uint16_t RED    = 4;
  // Ultrasonic
    const int us_max_dist = 50;  // 500 mm == 50 cm
    const int max_sensor_distance = 500;
    const int min_sensor_distance = 200;
  // Running Loop
    bool running;
  // Temporary Line Loss
    int lostFrames = 0;
    const int MAX_LOST_FRAMES = 8; // Allows brief loss of line without stopping
    int storedVar = 0;
    int storedError = 0;
  // PID Controller Parameters
    float Kp = 750;    // Proportional Gain  : directly scales current error (large value =
overshoot)
```

```
                //                      faster system response -> overshooting
    float Ki = 2;      // Integral Gain      : accumulates the error over time, adjusts for small errors
                //                      removes steady-state error -> overshooting
    float Kd = 0.1;    // Derivative Gain    : calculates the rate of change of error, dampens
left2right movement
                //                      improves stability, reduce overshooting -> more sensative to noise

NewPing us_sensor(us_trig_pin, us_echo_pin, us_max_dist);
Pixy2 pixy;
PIDLoop headingPID(Kp, Ki, Kp, false);

void setup() {
  // Initialize Serial
    Serial.begin(9600);
  // Initialize Pixy
    pixy.init();
    pixy.setLamp(1, 1);
    pixy.changeProg("color_connected_components");
  // Initialize Pins
    // Limit Switch Pins
      pinMode(limit_switch_1, INPUT_PULLUP);
      pinMode(limit_switch_2, INPUT_PULLUP);
    // Motor Pins
      pinMode(LeftMotor_PWM,  OUTPUT);
      pinMode(LeftMotor_EN,   OUTPUT);
      pinMode(RightMotor_PWM, OUTPUT);
      pinMode(RightMotor_EN,  OUTPUT);
    // LED Pins
      pinMode(rearLed , OUTPUT);
      pinMode(rgbRed  , OUTPUT);
      pinMode(rgbGreen, OUTPUT);
      pinMode(rgbBlue , OUTPUT);

  // Set initial LED state
    setLEDState(0);
  // Initialize Motors
    stopMotors();
  // Initialize Running Loop
    running = true;
  // Tilt the Pixy Down
    pixyDown();
}

void loop() {
  if (running)
  {
```

```
checkLimitSwitch();

// Initialie Variables
  int error = 0;
  int adj_error = 0;

// Scan For Blocks
  pixy.setLamp(1, 1);
  // pixy.setCameraBrightness(60);
  int8_t numBlocks = pixy.ccc.getBlocks();

// If A Block Is Detected
if (numBlocks > 0) {
  // Initialize Variables
    int selectedBlockIndex = -1;
    int smallestDelta = 1000;
  // For Every Block Detected
  for (int i=0; i < numBlocks; i++) {
    int sig = pixy.ccc.blocks[i].m_signature;
    if (sig == BLUE) {
      // Get Center X Value For Sig & Calculate The Absolute Error
        int centroid_x = pixy.ccc.blocks[i].m_x;
        int delta = abs(centroid_x - (pixy.frameWidth/2));
      // Store And Update The Selected Block Based On Smallest Delta
      if (delta < smallestDelta) {
        smallestDelta = delta;
        selectedBlockIndex = i;
      }
    }
  }
  // Once Block Has Been Selected
  if (selectedBlockIndex != -1) {
    // Check If It's A Stop Line
      bool stopLine = checkStopLine(selectedBlockIndex);
    // If A Stop Line Is Detected
    if (stopLine == true) {
      // Stop The Motors
        stopMotors();
      // Determine And Process Intersection
        determineIntersection();
    } else {  // Proceed With Normal Line Following
      // Calculate Heading Error & Update PID
        int centroidX = pixy.ccc.blocks[selectedBlockIndex].m_x;
        error = centroidX - (pixy.frameWidth / 2);
        headingPID.update(error);
        int adj_correction = headingPID.m_command / 2;
        // Print to Serial Monitor For Debugging
          Serial.print("Centroid X: ");
          Serial.print(centroidX);
          Serial.print(" | Error: ");
          Serial.print(error);
          Serial.print(" | PID Output: ");
          Serial.println(adj_correction);
        // Control The Motors Accordingly
          handleMovement(adj_correction, error);
        // Store Current Values In Case Of Frame Loss
          storedVar   = adj_correction;
          storedError = error;
          lostFrames = 0;
        }
      } else {
       // No Blue Line Is Being Detected
         Serial.println("No blue line detected... Resuming from last input.");
       // Lost Frame Mechanism
       if (lostFrames < MAX_LOST_FRAMES) {
         // If Line Is Lost, Control Motors Using Last Known Input
           handleMovement(storedVar, storedError);
           lostFrames++;
       } else {
         return;
       }
       // If Line Is Not Found After Continuing Prior Input
       Serial.println("Blue Line Is Still Not Detected.. Stopping.");
       stopMotors();
      }
    // Slight Delay To Reduce Computational Load
    //delay(100);
    }
  }
else {
    checkLimitSwitch();
  }
}

// Function to check the state of the limit switch
void checkLimitSwitch(void) {
  if (running)
  {
    // If limit switch is pressed
    if (digitalRead(limit_switch_1) || digitalRead(limit_switch_2) == LOW)
    {
      Serial.println("Collision Detected!! Stopping the motors & entering state 2...");
    stopMotors();   // Stop Motors
    setLEDState(2); // Enter Collision State
    running = false;
   }
  } else
   {
    Serial.println("Waiting for Manual Reset!");
    delay(1000);
   }
}

// Function to read the distance from the US and calculate the scalar for the motor speed
int ultrasonicDistanceLogic (void) {
  int distance_cm, distance_mm;
  float scalar;
  distance_cm = us_sensor.ping_cm();
  distance_mm = (distance_cm * 10);

  if ((distance_mm < min_sensor_distance) | (distance_mm = min_sensor_distance))  // If Less
Than or Equal to 200mm
   {
    scalar = 0;
   }
  else if ((distance_mm > max_sensor_distance) | (distance_mm = max_sensor_distance)) // If
Greater Than or Equal to 500mm
   {
    scalar = 1;
   }
  else // If Distance is between (200mm - 500mm)
   {
    scalar = float(distance_mm - min_sensor_distance) / (max_sensor_distance -
min_sensor_distance);
    scalar = constrain(scalar, 0.00, 1.00);
   }

  return scalar;
}

// Stop Motor Function
// Def:   Disables The Motors And Sets PWM To 0
void stopMotors() {
  analogWrite(LeftMotor_PWM,    0);
  analogWrite(RightMotor_PWM,   0);
  digitalWrite(LeftMotor_EN,    LOW);
  digitalWrite(RightMotor_EN,   LOW);
}
```

```
// Move Forward Function
// Def:   Moves Vehicle Forward From Stopline (used for intersections)
void moveForward() {
  // Call handleMovement Function
  // Correction = 0 , Error = 0
    const int baseSpeed   = 85;
    const int minSpeed    = 80;
    const int maxSpeed    = 150;
      // Calculate & Set Motor Speeds
      int leftSpeed, rightSpeed = baseSpeed;
      int corr          = leftSpeed;
      float ratio       = (3.76*pow(10, -8))*pow(corr, 3) + (-2.07*pow(10, -5))*pow(corr, 2) +
(4.57*pow(10, -3))*corr + 0.608;
      int adj_leftSpeed = leftSpeed*ratio;
      adj_leftSpeed     = constrain(adj_leftSpeed, minSpeed, maxSpeed);
  // Enable Motors
    digitalWrite(LeftMotor_EN,    HIGH);
    digitalWrite(RightMotor_EN,   HIGH);
  // Set Motor Speeds
    analogWrite(LeftMotor_PWM,    adj_leftSpeed);
    analogWrite(RightMotor_PWM,   rightSpeed);

    delay(1500);  // 1.75 seconds
    stopMotors();
}

// Turn Left Function
// Def:   Makes Vehicle Complete A 90 Degree Turn To The Left
void turnLeft() {
  // Serial Prints
    Serial.println("Called turnLeft Function!");
    pixyDown();
    pixy.setLamp(1, 1);
    delay(1000);
  // Enable Motors
    digitalWrite(LeftMotor_EN,    HIGH);
    digitalWrite(RightMotor_EN,   HIGH);
  // Set Motor Speeds
    analogWrite(LeftMotor_PWM,    0);
    analogWrite(RightMotor_PWM,   85);
    delay(300);
      // Wait Until Line Is In Center
        bool isCentered = false;
        while (!isCentered) {
          // Scan for Blue Sig
          int sig     = pixy.ccc.getBlocks(true, uint8_t(BLUE));
        // Get Position Values
          int sig_x   = pixy.ccc.blocks[0].m_x;
        // Determine Error
          int center_error      = abs(sig_x - (pixy.frameWidth / 2));
          int abs_center_error  = abs(center_error);
          isCentered  = abs_center_error < 25;
          Serial.print("isCentered: ");
          Serial.print(isCentered);
          Serial.print(" | sig_x = ");
          Serial.print(sig_x);
          Serial.print(" | abs_error = ");
          Serial.println(abs_center_error);
      }
      stopMotors();
}

// Turn Right Function
// Def:   Makes Vehicle Complete A 90 Degree Turn To The Right
void turnRight() {
  // Serial Prints
    Serial.println("Called turnRight Function!");
    pixyDown();
    pixy.setLamp(1, 1);
    delay(1000);
  // Enable Motors
    digitalWrite(LeftMotor_EN,    HIGH);
    digitalWrite(RightMotor_EN,   HIGH);
  // Set Motor Speeds
    analogWrite(LeftMotor_PWM,    85);
    analogWrite(RightMotor_PWM,   0);
    delay(300);
      // Wait Until Line Is In Center
      bool isCentered = false;
      while (!isCentered) {
        // Scan for Blue Sig
          int sig     = pixy.ccc.getBlocks(true, uint8_t(BLUE));
        // Get Position Values
          int sig_x   = pixy.ccc.blocks[0].m_x;
        // Determine Error
          int center_error      = abs(sig_x - (pixy.frameWidth / 2));
          int abs_center_error  = abs(center_error);
          isCentered  = abs_center_error < 25;
          Serial.print("isCentered: ");
          Serial.print(isCentered);
          Serial.print(" | sig_x = ");
```

```
            Serial.print(sig_x);
            Serial.print(" | abs_error = ");
            Serial.println(abs_center_error);
        }
        stopMotors();
}

// Pixy Down Function
// Def:   Adjusts Pixy2 Servos To Make It Look Down
void pixyDown() {
  // Set The Servos (PAN , TILT)
  pixy.setServos(445, 975);
}

// Pixy Up Function
// Def:   Adjusts Pixy2 Servos To Make It Look Up
void pixyUp() {
  // Set The Servos (PAN , TILT)
  pixy.setServos(445, 500);
}

// Intersection Determination Function
// Def:   Raises the Pixy2 And Checks For A Directional Device
void determineIntersection() {
  // Serial Prints
    Serial.println("Intersection Detected... Determining Availability Of Directional Device...");
  // Initialize Variables
    bool dirDeviceDetected  = false;
    int16_t yellow_x        = 0;
    int16_t yellow_y        = 0;
    int16_t selected_yellow_x   = 0;
    int16_t selected_yellow_y   = 0;
    bool temp = false;
  // Raise The Pixy2 & Allow It To Stabalize
    pixyUp();
    pixy.setLamp(0, 0);
    //pixy.setCameraBrightness(30);
    delay(1500);
  // Scan For Yellow Signature A Few Times
    int k = 0;
    int8_t yellowBlocks   = pixy.ccc.getBlocks(true, uint8_t(YELLOW));
    for (k = 0; k < 5; k++) {
      yellowBlocks   = pixy.ccc.getBlocks(true, uint8_t(YELLOW));
    }
  // If Yellow Was Detected
  if (yellowBlocks > 0) {
```

```
  for (int i = 0; i < yellowBlocks; i++) {
   // Signature Check
   uint8_t sig = pixy.ccc.blocks[i].m_signature;
   if (sig == YELLOW) {
    // Store Positional Values Of Yellow Signature
     yellow_x  = pixy.ccc.blocks[i].m_x;
     yellow_y  = pixy.ccc.blocks[i].m_y;
   }
   bool yellow_x_check   = (yellow_x != 0);
   bool yellow_y_check   = (yellow_y != 0);
   Serial.print("Yellow X Check: ");
   Serial.println(yellow_x_check);
   Serial.print("Yellow Y Check: ");
   Serial.println(yellow_y_check);

   if (yellow_x_check && yellow_y_check) {
    // Threshold Check
    bool horizontal   = ((yellow_x >= 100) && (yellow_x <= 200));
    bool vertical     = ((yellow_y >= 100) && (yellow_y <= 200));

    Serial.print("Horizontal Check: ");
    Serial.println(horizontal);
    Serial.print("Vertical Check: ");
    Serial.println(vertical);
    if (vertical && horizontal) {
     selected_yellow_x   = yellow_x;
     selected_yellow_y   = yellow_y;
     // There is A Directional Device Present
      dirDeviceDetected = true;
      temp = true;
    }
   }
 }
 }
 // Determine How To Respond
 if (dirDeviceDetected) {      // (yellow_x != 0)
  Serial.println("Directional Device Detected... Processing Intersection");
  Serial.print("Dectected Position of Yellow (x , y):  (");
  Serial.print(yellow_x);
  Serial.print(" , ");
  Serial.print(yellow_y);
  Serial.println(") ");
  processIntersection_DD(yellow_x, yellow_y);
 } else {
  Serial.println("No Directional Device Present... Processing Intersection");
  processIntersection_noDD();
  }

}
// Non-Directional Device Intersection Function
// Def:   Controls Yielding & Handling Of Non DD Intersections
void processIntersection_noDD() {
  // First, Check For Other Cars And Wait For Clear Yield Status
    Serial.println("Checking For Other Vehicles...");
    bool yield_status = false;
    while (!yield_status) {
      yield_status = yieldCheck();
    }
  // Once Yield Status Is Clear
    Serial.println("Yield Status Clear!!! Continuing...");
  // Approach Intersection
    moveForward();
  // Check Directions (LFT/FRWD/RGT) To Determine Available Routes
    Serial.println("Checking Best Direction...");
    int selected_direction = checkBestDirection();  // -1 Left, 0 Forward, 1 Right
    pixyDown();
    delay(2000);
      // Process Direction
    if (selected_direction == -1) {         // Left
      // Turn Left
        turnLeft();
        return;
    } else if (selected_direction == 0) {   // Forward
      // Go Straight
        moveForward();
        return;
    } else if (selected_direction == 1) {   // Right
      // Turn Right
        turnRight();
        return;
    } else {
      // Error Case - Didnt Determine Direction
        Serial.println("ERROR - Didnt Determine Direction From Signature Area...");
        stopMotors();
        delay(10000);
    }
}

// Yield Status Function
// Def:   Determines If Vehicle Has To Yield Or Has Right Of Way, Returns "All-Clear"
bool yieldCheck() {
  // Initialize Variables
  int currentHeight       = -1;
  int previousHeight      = -1;
  int previous_x          = -1;
  int current_x           = -1;
  int16_t green_x         = -1;
  bool red_detected       = false;
  const int time_limit    = 2500;   // 5 seconds
  int time_counter        = 0;
  bool car_on_left        = false;
  bool car_on_right       = false;
  const int centerFrame   = pixy.frameWidth/2;
  bool vehicle_detected   = false;
// Initial Scan
  // Scan For Green & Then Scan For Red Signatures
    int green_sigs  = pixy.ccc.getBlocks(true, uint8_t(GREEN));
  // If A Green Sig Is Detected
  if (green_sigs > 0) {
    for (int i = 0; i < green_sigs; i++) {
      uint8_t sig = pixy.ccc.blocks[i].m_signature;
      if (sig == GREEN) {
        vehicle_detected  = true;
        green_x  = pixy.ccc.blocks[0].m_x;
      }
    }
  }
  // Scan For Red
    int red_sigs      = pixy.ccc.getBlocks(true, uint8_t(RED));
  // If A Red Sig Is Detected
  if (red_sigs > 0) {
    for (int i = 0; i < red_sigs; i++) {
      uint8_t sig2 = pixy.ccc.blocks[i].m_signature;
      if (sig2 == RED) {
        red_detected      = true;
        currentHeight     = pixy.ccc.blocks[0].m_height;
        current_x         = pixy.ccc.blocks[0].m_x;
      }
    }
  }
// If Only Green Is Detected
if (vehicle_detected && !red_detected) {
  // Determine Which Side
  // Serial Prints
    Serial.println("Front Of Other Vehicle Detected, Determining Right Of Way...");
  // Use Most Significant Green Sig & Determine Its Position Relative To Our Vehicle
    car_on_left       = green_x < centerFrame;
    car_on_right      = green_x > centerFrame;
    // If Another Vehicle Is Present
    if (car_on_left) {    // Yield
      // Serial Prints
        Serial.println("Vehicle Detected To Our Left..");
      // Quick Delay, Then Continue To Scan For Red Signatures
        delay(200);
        bool clear  = false;
        // Loop To Ensure That Yield Is Complete
        while (!clear) {
          clear = scanForRed();
        }
        return true;
    } else if (car_on_right) {    // Do Not Yield
      // Serial Prints
        Serial.println("Vehicle Detected To Our Right..");
      // We Have Right Of Way, Return The All Clear
        return true;          // Exit Function -- Return (1)
    }
  }
  // If Red Is Detected At All
  if (red_detected) {
    bool clear = false;
    while (!clear) {
      clear = scanForRed();
    }
    return true;
  }
  // If Neither Signature Is Detected
  if (!vehicle_detected && !red_detected) {
    // Automatically Good To Approach Intersection
      return true;
  }
}

// Check Best Direction Function
// Def:   Controls The Servos To Scan Multiple Paths And Determine The Correct Route
int checkBestDirection() {
  // First Check Availability Of Forward Direction
    pixy.setLamp(1, 1);
    int forwardArea     = getBlockArea();
  // If Forward Is Available
  if (forwardArea > 0) {
    // Serial Prints
      Serial.println("Forward Path Detected. Returning '0'.");
    return 0;
  }
```

```
 // Forward Path Not Available, Check LFT/RGT Areas
 // Serial Prints
   Serial.println("Forward Path Not Available... Checking Left & Right Paths.");
 // Checking & Storing Left Area
   pixy.setServos(750, 775);
   delay(2000);    // Stabilization
   int leftArea      = getBlockArea();
 // Checking & Storing Right Area
   pixy.setServos(200, 775);
   delay(2000);    // Stabilization
   int rightArea     = getBlockArea();
 // Determining Path
 if (leftArea > rightArea) {
   // Serial Prints
     Serial.print("Turning Left! Block Area Comparison: ");
     Serial.print(leftArea);
     Serial.print(" > ");
     Serial.println(rightArea);
   // Path Selected, Returning '-1'
   return -1;
 } else if (leftArea < rightArea) {
   // Serial Prints
     Serial.print("Turning Right! Block Area Comparison: ");
     Serial.print(leftArea);
     Serial.print(" < ");
     Serial.println(rightArea);
   // Path Selected, Returning '1'
   return 1;
 }
}

// Get Block Area Function
// Def:   Scans For The Most Significant Block & Returns Its Area (blue sig only)
int getBlockArea() {
 // Variables
   int totalArea     = 0;
   int width, height = 0;
 // Scan For Blue Signatures
   int blue_sigs     = pixy.ccc.getBlocks(true, uint8_t(BLUE));
 // Gather Height And Width Values
   width     = pixy.ccc.blocks[0].m_width;
   height    = pixy.ccc.blocks[0].m_height;
   totalArea = width * height;
 // Return The Total Area
 return totalArea;
```

```
}

// Scan For Red Function
// Def:   Function That Continuously Scans For Red Signature Changes
bool scanForRed() {
  // Variables, Constants, Bools
    bool clear          = false;
    int currentHeight   = -1;
    int previousHeight  = -1;
    int current_x       = -1;
    int previous_x      = -1;
    int selectedBlock   = -1;
    const int frame_max = 285;
    const int frame_min = 25;
    const int cnt_max   = 5;
    int yld_count       = 0;
  while (!clear) {
    // Scan For Red Signatures
      int red_sig       = pixy.ccc.getBlocks();
    // If A Red Signature Was Detected
    if (red_sig > 0) {
      for (int i=0; i<red_sig;i++) {
        uint8_t sig = pixy.ccc.blocks[i].m_signature;
        if (sig == 0b00001000) {
          currentHeight = pixy.ccc.blocks[i].m_height;
          current_x     = pixy.ccc.blocks[i].m_x;
          selectedBlock = i;
        }
      }
      if (selectedBlock != -1) {
        // Serial Prints
          Serial.println("Red Signature Detected...");
        // Compare To Previous Values
          bool gettingSmaller   = (previousHeight != -1) && (currentHeight < previousHeight);
          bool crossingInt      = (abs(current_x - previous_x) != 0);
          bool exitingFrame     = (current_x >= frame_max);
          if (gettingSmaller || exitingFrame || crossingInt) {
            Serial.println("Increasing Yield Count..");
            yld_count++;
          } else {
            yld_count = 0;
            Serial.println("Red Light Disappeared... Resetting Yield Count.");
          }
        // Counter For Multiple Frame Check
          if (yld_count == cnt_max) {
            // Yield Is Complete
          Serial.println("Yield Is Complete! Returning...");
          return true;
        }
      }
    }
  }
}

// Directional Device Intersection Function
// Def:   Completes A Turn (LEFT / RIGHT) Based On GRN/YLW Signature Locations
void processIntersection_DD(int16_t yellow_x, int16_t yellow_y) {
  // Initialize Variables & Constants
    const int16_t y_threshold          = 20;
    int16_t green_x, selected_green_x   = 0;
    int16_t green_y, selected_green_y   = 0;
    bool greenDetected  = false;
    uint8_t comp = -1;
    int k = 0;
  // We Have Yellow LEDs Position, So Find Green
    int8_t greenBlocks  = pixy.ccc.getBlocks();
    for (k = 0; k < 5; k++) {
        int8_t greenBlocks  = pixy.ccc.getBlocks();
    }
  // If Green was detected
  if (greenBlocks > 0) {
    for (int i = 0; i < greenBlocks; i++) {
      // Signature Check
        uint8_t sig = pixy.ccc.blocks[i].m_signature;
      if (sig == GREEN) {
        // Store Positional Values of Green Signature
          green_x   = pixy.ccc.blocks[i].m_x;
          green_y   = pixy.ccc.blocks[i].m_y;
        // Serial Monitor Prints
          Serial.print("Green X: ( ");
          Serial.print(green_x);
          Serial.print(" )  Green Y: ( ");
          Serial.print(green_y);
          Serial.println(" )");
      }
      // Threshold Check
      bool horizontal   = ((green_x >= 75) && (green_x <= 250));
      bool vertical     = ((green_y >= 100) && (green_y <= 200));

      if (horizontal && vertical) {
        selected_green_x  = green_x;
        selected_green_y  = green_y;
        greenDetected     = true;
      }
    }
  }
    if (greenDetected) {
      // Bools For Comparisons
        bool signalLeft         = selected_green_x < yellow_x;
        bool signalRight        = selected_green_x > yellow_x;
      // Direction Confirmation & Response
      if (signalLeft) {
        // Green Sig Is LEFT Of Yellow Sig
        Serial.println("Turning Left!!");
        moveForward();
        turnLeft();
        return;
      } else if (signalRight) {
        // Green Sig Is RIGHT Of Yellow Sig
        Serial.println("Turning Right!!");
        moveForward();
        turnRight();
        return;
      } else {
        // Error - Didnt Pick Up Left Or Right
          Serial.println("Scanned A Green Signature But Didn't Obtain A Turn Direction...");
      }
    } else {
      Serial.println("Green Not Detected!");
    }
}

// Stop Line Detection Function
// Def:   Checks If Currently Selected Block Is A Stop Line
bool checkStopLine(int selectedBlock) {
  // Stop Line Constants
    const int stop_line_width_threshold   = 140;
    //const int stop_line_width_limit       = 195;
    //const int stop_line_height_threshold  = 100;
    const int stop_line_y_threshold       = 125;
    //const int stop_line_x_threshold       = ;
    const int stop_line_center_x          = pixy.frameWidth / 2;
    const int stop_line_center_margin       = pixy.frameWidth * 0.1;  // 10% Margin
  // Initialize Variables
    bool stopLineDetected = false;
    bool blockCentered    = false;
    bool blockLow         = false;
    bool blockWide        = false;
```

```
   bool blockHeightComp  = false;
  // Gather Positional Data Of Signature
   int blockWidth    = pixy.ccc.blocks[selectedBlock].m_width;
   int blockHeight   = pixy.ccc.blocks[selectedBlock].m_height;
   int block_x_pos   = pixy.ccc.blocks[selectedBlock].m_x;
   int block_y_pos   = pixy.ccc.blocks[selectedBlock].m_y;
   // Serial Prints
     Serial.print("Block ");
     Serial.print(selectedBlock);
     Serial.print(" - Width: ");
     Serial.print(blockWidth);
     Serial.print(" | Height: ");
     Serial.print(blockHeight);
     Serial.print(" | X: ");
     Serial.print(block_x_pos);
     Serial.print(" | Y: ");
     Serial.println(block_y_pos);
  // Set Up Bools
   blockCentered     = abs(block_x_pos - stop_line_center_x) <= stop_line_center_margin;
   blockLow          = (block_y_pos >= stop_line_y_threshold);
   blockWide         = (blockWidth >= stop_line_width_threshold);
   //blockHeightComp   = (blockHeight >= stop_line_height_threshold);
  // Print Bool Status
   Serial.print("Block Centered: ");
   Serial.print(blockCentered);
   Serial.print(" | Block Low: ");
   Serial.print(blockLow);
   Serial.print(" | Block Wide: ");
   Serial.println(blockWide);
  // Serial.print(" | Block Height Comp: ");
  // Serial.println(blockHeightComp);
  // Check If Stop Line
  if (blockCentered && blockLow && blockWide) {
   // Serial Print
     Serial.println("Stop Line Detected!");
   stopLineDetected    = true;
   return stopLineDetected;
  } else {
   stopLineDetected    = false;
   return stopLineDetected;
  }
}

// Handle Movement Function
// Def:   Used To Control The Speeds Of The Motors
void handleMovement(int correction, int error) {
```

```
// Initialize Variables
  const int baseSpeed   = 85;
  const int minSpeed    = 80;
  const int maxSpeed    = 150;
  const int pivotSpeed  = 80;
  const int pivotThreshold = 20;
  const int widthThreshold = 50;
  const int heightThreshold = 50;
  int width   = pixy.ccc.blocks[0].m_width;
  int x_pos   = pixy.ccc.blocks[0].m_x;
  int height  = pixy.ccc.blocks[0].m_height;
  //bool leftRightLimit   =
  bool widthCheck   = (width >= widthThreshold);
  bool heightCheck  = (height <= heightThreshold);
  bool x_pos_check  = abs(error) > pivotThreshold;
  Serial.print("widthCheck: ");
  Serial.print(widthCheck);
  Serial.print(" | x_pos_check: ");
  Serial.println(x_pos_check);
// If Error Exceeds The Pivot Threshold
if (x_pos_check && widthCheck && heightCheck) {
  Serial.println("Sharp Turn Detected -> Entering Pivot Mode");
  // Determine Pivot Direction (- Error = Left, + Error = Right)
  if (error < 0) {
    Serial.println("Pivoting Left...");
    digitalWrite(LeftMotor_EN,  HIGH);
    digitalWrite(RightMotor_EN, HIGH);
    analogWrite(LeftMotor_PWM,  0);
    analogWrite(RightMotor_PWM, pivotSpeed);
    // Wait Until Line Is In Center
      bool isCentered = false;
      while (!isCentered) {
        // Scan for Blue Sig
          int sig     = pixy.ccc.getBlocks(true, uint8_t(BLUE));
        // Get Position Values
          int sig_x   = pixy.ccc.blocks[0].m_x;
        // Determine Error
          int center_error      = abs(sig_x - (pixy.frameWidth / 2));
          int abs_center_error  = abs(center_error);
          isCentered  = abs_center_error < 25;
          Serial.print("isCentered: ");
          Serial.print(isCentered);
          Serial.print(" | sig_x = ");
          Serial.print(sig_x);
          Serial.print(" | abs_error = ");
          Serial.println(abs_center_error);
      }
    } else {
      Serial.println("Pivoting Right...");
      digitalWrite(LeftMotor_EN,  HIGH);
      digitalWrite(RightMotor_EN, HIGH);
      analogWrite(LeftMotor_PWM,  pivotSpeed);
      analogWrite(RightMotor_PWM, 0);
      // Wait Until Line Is In Center
        bool isCentered = false;
        while (!isCentered) {
          // Scan for Blue Sig
            int sig     = pixy.ccc.getBlocks(true, uint8_t(BLUE));
          // Get Position Values
            int sig_x   = pixy.ccc.blocks[0].m_x;
          // Determine Error
            int center_error      = abs(sig_x - (pixy.frameWidth / 2));
            int abs_center_error  = abs(center_error);
            isCentered  = abs_center_error < 20;
            Serial.print("isCentered: ");
            Serial.print(isCentered);
            Serial.print(" | sig_x = ");
            Serial.print(sig_x);
            Serial.print(" | abs_error = ");
            Serial.println(abs_center_error);
        }
    }
    // Movement Has Been Handled, Return To Beginning Of Loop
    return;
  }
  // If Error Is Within The Pivot Threshold
    // Initialize & Adjust L/R Motor Speeds
      int leftSpeed   = constrain(baseSpeed + correction, minSpeed, maxSpeed);
      int rightSpeed  = constrain(baseSpeed - correction, minSpeed, maxSpeed);
    // Serial Prints
      if (error > 0) {
        Serial.println("Turning RIGHT");
      } else if (error < 0) {
        Serial.println("Turning LEFT");
      } else {
        Serial.println("Going STRAIGHT");
      }
    // Calculate & Set Motor Speeds
      int corr          = leftSpeed;
      float ratio       = (3.76*pow(10, -8))*pow(corr, 3) + (-2.07*pow(10, -5))*pow(corr, 2) +
(4.57*pow(10, -3))*corr + 0.608;
      int adj_leftSpeed = leftSpeed*ratio;
      adj_leftSpeed     = constrain(adj_leftSpeed, minSpeed, maxSpeed);

      digitalWrite(LeftMotor_EN,  HIGH);
      digitalWrite(RightMotor_EN, HIGH);
      analogWrite(LeftMotor_PWM,  adj_leftSpeed);
      analogWrite(RightMotor_PWM, rightSpeed);
      // Serial Prints
        Serial.print("Left Speed: ");
        Serial.print(adj_leftSpeed);
        Serial.print(" | Right Speed: ");
        Serial.println(rightSpeed);
}

// Function to control LED states
void setLEDState(int state) {
  switch (state) {
    case 0: // S0 - Initial state
      digitalWrite(rearLed, LOW);
      digitalWrite(rgbRed, LOW);
      digitalWrite(rgbGreen, LOW);
      digitalWrite(rgbBlue, LOW);
      break;
    case 1: // S1 - Operating state
      digitalWrite(rearLed, HIGH);
      digitalWrite(rgbRed, LOW);
      digitalWrite(rgbGreen, HIGH);
      digitalWrite(rgbBlue, LOW);
      break;
    case 2: // S2 - Collision state
      digitalWrite(rearLed, HIGH);
      digitalWrite(rgbRed, HIGH);
      digitalWrite(rgbGreen, LOW);
      digitalWrite(rgbBlue, LOW);
      break;
    case 3: // S3 - Detection error mode
      digitalWrite(rearLed, HIGH);
      digitalWrite(rgbRed, LOW);
      digitalWrite(rgbGreen, LOW);
      digitalWrite(rgbBlue, HIGH);
      break;
  }
}
```