SOAP BOX DERBY TUNING OPTIMIZATION

a report done for

Mr. Bob Getts
Indianapolis Soap Box Derby
Indianapolis, Indiana

by

Charles Earle
Joshua Eckels
Pascal Liberge
Shane Saylor

May 11, 2021

In partial fulfillment of the requirements of the course

ME 472
Senior Design
Professor Zachariah Chambers

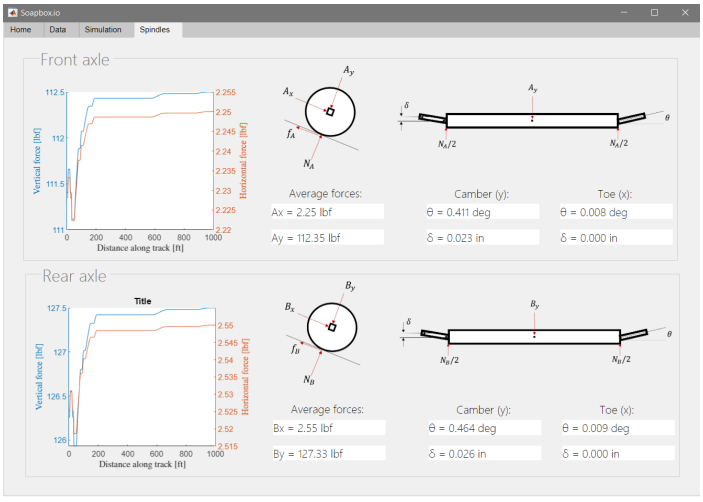
DEPARTMENT OF MECHANICAL ENGINEERING
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Executive Summary

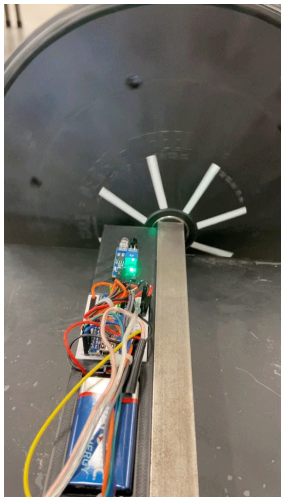
Client background and needs: The primary client for this project is Bob Getts, the director of the Indianapolis Soap box Derby Organization. Soap box derby is a youth car racing competition, where participants race gravity-powered soap box cars down a hill. There are two main factors that affect race performance: pre-race car tuning, and the skill of the driver. The client wants to improve his team's performance by optimizing the pre-race car tuning process. The client needs a tool that determines optimal car tunings based on racing conditions. The tool needs to be quick-to-use, user-friendly, scalable, and require minimal data collection. The tool should not require expensive software licenses. Our ability to meet these needs will be evaluated by race time improvement, and by processing and data collection times.

Design solution: The final design is a two-pronged solution approach to the problem. The first prong is a software program that takes input data from the client and displays relevant output tuning settings. The program will account for the race track hill profile, the weight of the driver, and the geometry of the car. The program will provide the client with optimal weight, camber, and toe conditions. The second prong of the solution is a velocity data-logging tachometer that measures the total resistive forces acting on the soapbox derby car. The client may use this device by making minor adjustments to a car and seeing how the changes affect the road load acting on the car. If the changes caused the road load to decrease, then the changes will boost the performance of the car during a race. Historically, these tuning settings have been determined experimentally by the client through trial and error. This design solution will replace the need for human trial and error and will adaptively provide optimal tuning settings for variable racing conditions.

Prototype: A prototype software tool was created as a proof-of-concept of the software design. The user is able to enter race parameters, such as the weight of the driver and the slope of the hill, and the program outputs optimal settings for the weight distribution in the car and axle toe and camber deflections. A prototype tachometer was also printed and assembled to the front axle of a soapbox derby car as shown in figure 1.



(a) Software prototype with spindle tuning settings and deflection results



(b) Velocity tachometer prototype

Figure 1: Prototypes for software tool and velocity tachometer

Nomenclature

The axle is a solid rectangular metal bar upon which the floorboard of a soap box car is attached via a king pin in the middle of the axle. The wheels are attached via bearings to the spindles, which are cylindrical extrusions on either end of the axle. The wheels rotate on the bearings during operation, and the axle/spindle bar remains fixed. An illustration of a soap box car axle is shown in figure 2.

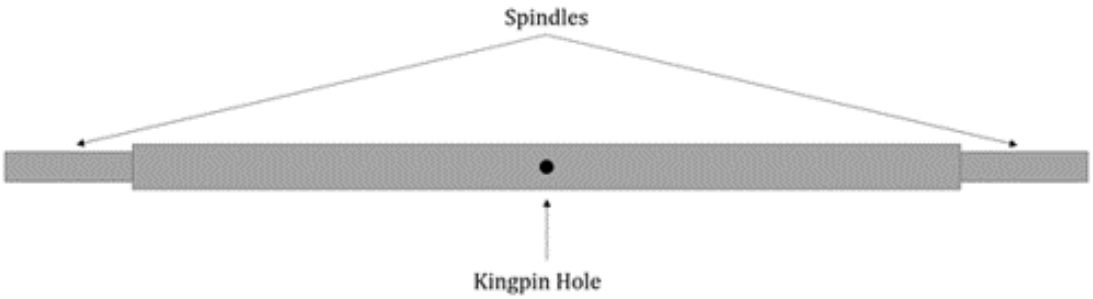


Figure 2: Illustration of soap box car axle

The floorboard is a flat wooden board that composes the base of the car. The driver sits on the floorboard, and additional weights can be placed on top of the floor board to move the center of gravity of the vehicle. A diagram of moveable weight locations on a typical soap box car floorboard is shown in figure 3.

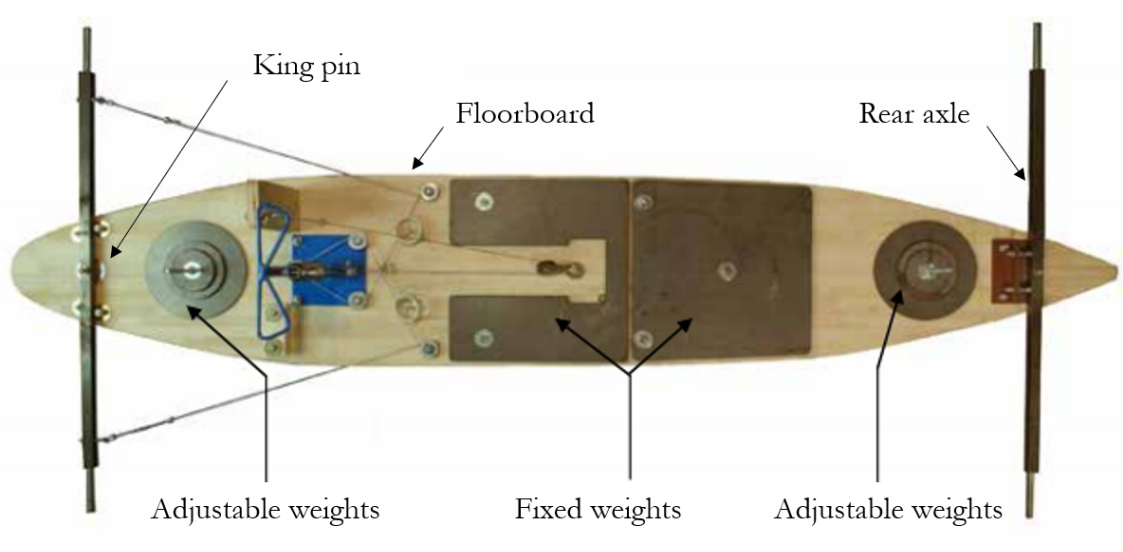


Figure 3: Diagram of moveable weights on floorboard of soap box car [1]

.

The shell is a lightweight body that rests on top of the floorboard and covers the driver, weights, and other mounted components. Some typical soap box car shells are shown in figure 4.

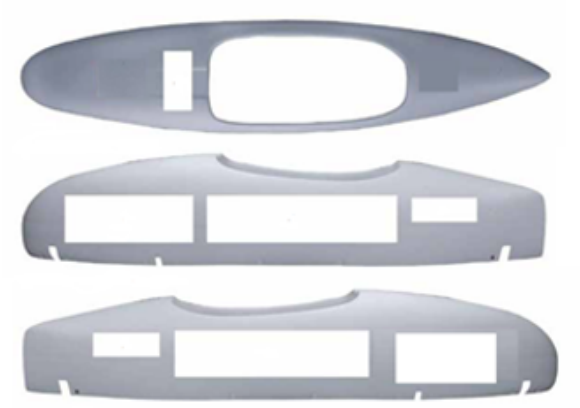
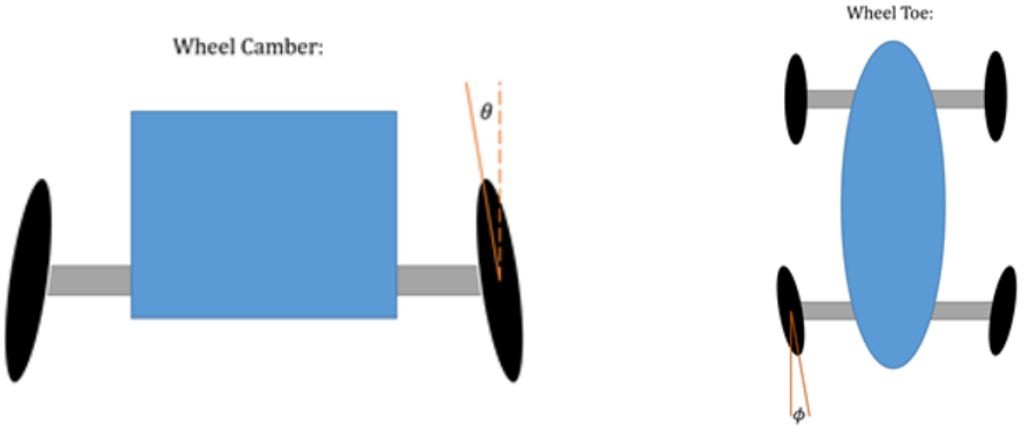


Figure 4: Typical soap box derby car shells [1]

.

The camber and toe angles of an axle assembly are shown in figure 5.



(a) Wheel camber angle θ (front view)

(b) Wheel toe angle ϕ (top view)

Figure 5: Camber and toe angles in soap box car

Disclaimer

The contents of this report were prepared by senior mechanical engineering students at Rose-Hulman Institute of Technology. We feel confident in our work as students. However, all material should be reviewed by an appropriate professional before implementation.

Contents

1 Problem definition

Soap Box Derby is a youth car racing program where kids construct small, gravity-powered soapbox cars and race them down a hill at derby competitions held across the country. Our client, Bob Getts, has a long history of competing in soap box derby races. He competed as a kid and now is passing his love for the sport to the Indianapolis community as the director of the Indianapolis Soap Box Derby. The sport is directed towards children from the ages of 7-18 and their families with the goal of "promoting families working together, teaching children some basic skills and sportsmanship through science and technology, teamwork, community service, family bonds, leadership, and diversity" [2]. A soap box race at Wilbur Shaw hill in Indianapolis is shown in figure 6.

(a) Wilbur Shaw soap box derby hill in Indianapolis, IN

(b) Two soap box car drivers racing downhill at Wilbur Shaw

Figure 6: Soap Box Derby Competition in Indianapolis, IN

To compete in these racing events, kids are given kits from which they can build and test their own soap box cars. The kits are designed so that kids show up on race day on a level playing field; none of the cars have a significant advantage over another in a given race. Success during the competition narrows down to two main factors: the tuning of the car immediately before a race, and the skill of the driver. This project focuses exclusively on the former.

Before a soap box race, the client will adjust the car to account for the weight of the driver and the conditions of the race track. The client will typically add weights to the car to shift the weight distribution either forward or backward in the car. The client will also bend the spindles to account for deflection in the axles, as well as adjust the tightness of the king pin to the center of the axle. Figure 7 shows some examples of the car tunings performed by the client before a typical soap box race.

(a) Weights are placed on the floorboard to move the overall center of gravity

(b) The end-to-end deflection of an axle is measured

Figure 7: Soap box tuning before a competition

The parameters suspected to have the largest impact on race time are the weight distribution and spindle deflection [3]. Through years of experience, the client typically determines the best value for these parameters experimentally to achieve a desired center of mass, as well as a desired camber and toe of the wheels. The goal of this project is to provide a more robust and automatic way to determine these optimal tuning settings over the current experimental method. In addition, the new method will significantly improve the performance of the Indianapolis soap box team by improving the accuracy of car tuning settings.

1.1 Problem statement

A model is needed to determine optimal weight, axle, and spindle tuning for soap box cars based on specifications about the racing conditions.

1.2 Project needs

The primary stakeholder for this project is Mr. Bob Getts, the director of the Indianapolis Soap Box Derby team. Mr. Getts was interviewed by the senior design team at the Wilbur Shaw Derby Hill to gain a better idea of the client's perspective. Notes from the interview are included in appendix A. The client needs a software program that can predict optimal soap box tuning settings from given environmental and driver conditions. The client wants the program to be free and easy to use. He also wants the program to act as a black box function, such that the program takes several input parameters, performs a series of back-end calculations, and displays actionable tuning settings. When applied to the car, these program-generated tuning settings will improve the car's performance during the race. The client plans to use the program several times on a single race day to account for different drivers and different cars, so he wants the processing time

to be quick. Secondary stakeholders for this project include the Indianapolis soap box derby team and the team's board of directors, as the success or failure of this project directly impacts their performance during soap box derby competitions.

After evaluating the perspectives of the client and stakeholders, the following list of project needs was developed:

1. **The program outputs feasible car tuning settings based on given input:** The program should be able to adapt to changing inputs. The outputs of the program should reflect changes in the input and should be within geometric, tooling, and other physical constraints. The output tuning settings for a given input should result in improved car performance over an otherwise unmodified car.

2. **The program is user-friendly:** It should be clear where input data is entered into the user interface and any output graphs or tuning settings should be obvious and comprehensible. It should require minimal training to use the program. The program should not require excessive effort from the user. The program should be thoroughly tested for errors and should handle incorrect user input by providing error and warning messages.

3. **Processing time should be quick:** The program should complete all modeling and calculation in a reasonable time. The user should be able to reliably use the program multiple times with minimal downtime for processing.

4. **Data collection should be simple and quick:** The program should not require extensive or complicated input from the user. Experimental data should be intuitive and quick to gather. Data collection should not require expensive measurement equipment. Any required equipment should be lightweight, inexpensive, transportable, intuitive, and easily obtainable. All data collection can be completed in the typical allotted time before a standard soap box derby race.

5. **The program should be easily scalable:** The program should be robust to future changes in client needs. The program should be implemented such that new data inputs can be easily accounted for should the need arise in the future. Future changes should be easy to implement with minimal effect on existing code; the code is modular.

6. **The program is free:** The final program should not require expensive software licenses for use. The final program delivered to the client should be in the form of a simple executable, requiring no further installation. There should be no associated costs for usage and distribution of the program.

1.3 Technical specifications

Based on the needs listed above, a list of technical specifications was compiled in order to evaluate how well the needs are met. Some of these specifications are goals that we consider to be feasible, and others are constraints that our program must accommodate in order to meet the client's needs. Most of these needs have straightforward metrics related to either the performance of the soap box derby car or the program itself. Each of the needs was ranked from 1 to 5 on level of importance to the success of the project. The needs and their corresponding specifications are shown in table 1.

Table 1: Ranked Technical Specifications and Targets

No.	Need	Rank	Metric	Unit	Target	Rationale
1	Feasible tuning settings	5	Race time improvement	sec	> 0.01	Margin of victory in typical race
2	User-friendly	4	User survey	1-10 scale	8/10	Client requirement
3	Quick processing time	3	Calculation time	min	< 1	Mid-race constraint
4	Easy data collection	3	Collection time	min	< 60	Pre-race constraint
5	Easily scalable	2	Binary	n/a	Yes	Client requirement
6	Free software	1	Binary	n/a	Yes	Client requirement

The target metrics were primarily determined via benchmarking conducted at the soap box derby race in Indianapolis. The first need states that the program should output feasible tuning settings. Since this is the client's primary motive for the project, this need was given the highest rank of 5 out of 5. Our ability to meet this need will be evaluated by how well the program can improve the race time of a soap box car. The target improvement time of 0.01s was selected because the margin of victory in soap box derby races is typically on the order of 0.001 to 0.01s. The next highest ranked need states that the program should be user-friendly. User-friendliness as a priority is required as this model would be of little help to our client if it was difficult or unintuitive to operate. This will be evaluated by how well our client ranks the project on a user survey.

The processing time and data collection of the project will be evaluated by a simple timed metric. The 60 minute data collection time was decided based on our interview with the client, where we learned that most teams spend roughly an hour on the day of the race preparing their cars and walking the track. As such, we would like all the data needed to run our model to be easily collected within this time frame. The client is also likely to use the program several times during an event, so the program should not take excessively long to process. Since the client has roughly 10 minutes between race heats to make any adjustments, the target value for processing time is less than 1 minute. The scalability and cost of the program require a simple binary metric of whether those needs were met or not. Because the client does not own a license to expensive engineering software, we will need to ensure usage of the program is free and does not require unnecessary expenditure.

The race time metric will be collected using timing software available through our client at Wilbur Shaw hill in Indianapolis. The calculation and data collection time will be measured with a simple timing watch, and a user survey will be provided to our client to rank the prototype on user-friendliness. The prototype will also be investigated if it meets the scalability and free software requirements.

2 Design solution

2.1 System overview

Our solution consists of a software tool that takes several input parameters and maps them to corresponding outputs and a tachometer that can record velocity data for the math model used by the software. The tachometer will be attached to the rear axle, replacing one of the aerodynamic fins to reduce the amount of variation between the soapbox car during races and during data collection. The software runs through several mathematical models to convert inputs such as vehicle geometry, which is constant within a class, and the client supplied inputs of vehicle weight, driver weight, and hill profile data, into outputs desired by the client: the amount of weight to add, the distribution of this weight, and the deflection of the spindles to produce desirable camber and toe angle for the wheels. A functional diagram is shown below.

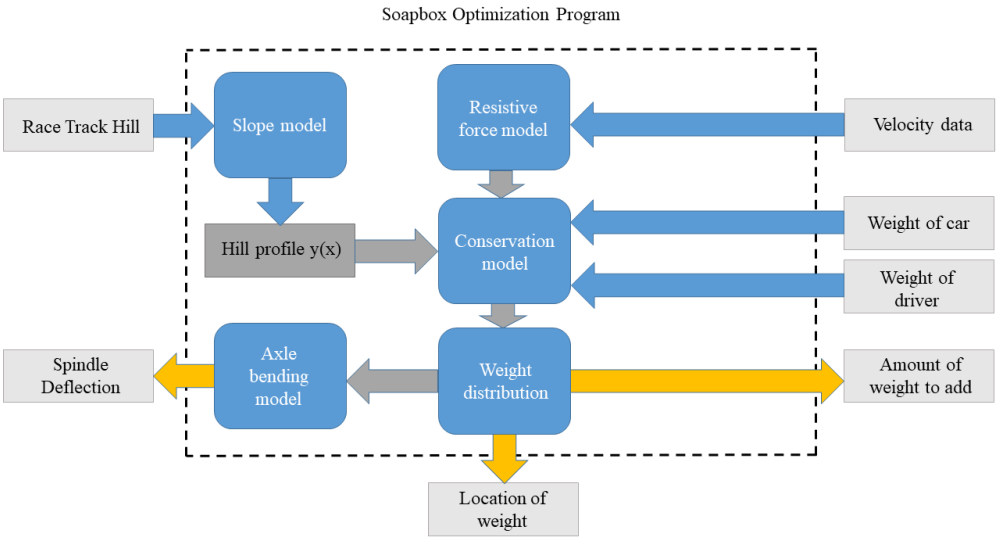


Figure 8: Functional diagram of the soapbox optimization program

The three inputs can be seen entering the system (blue arrows) and the corresponding outputs exiting the system (orange arrows). Each of the central blue chart elements represent a location in the flow of data where some form of appropriate system or dynamic modeling takes place. Each gray box or arrow represents an intermediate result. Each blue modeling element is described in further detail:

- Slope model: The profile of the race track will vary depending on the competition venue. The slope model is responsible for taking the profile of the race track into account. The profile of a given track can be approximated by collecting several slope measurements along the length of the track. The model approximates the curvature of the hill at every measurement point to construct a full hill profile. The hill profile measurement procedure is outlined in appendix B.
- Conservation model: Intuitively, the weight of the driver and car are input into the system through the application of an appropriate conservation model, as the soap box car is powered down the hill by gravity alone. Following previous work by Mann et al. [4] and Driscoll et al. [5], the two models under consideration for this project are momentum and energy

conservation. A conservation of momentum approach would focus on minimizing the race time, while conservation of energy would focus on minimizing losses in transit down the hill. The output of both conservation models would indicate the ideal weight distribution of the soap box car, given the weight of the driver and car and the profile of the hill, with all other factors held constant.

- Weight distribution: The conservation model will indicate the ideal weight distribution of the soap box car; however, there are limitations in how weight can be placed in the car. A weight distribution model is needed to determine the best arrangement of weights in the car to most closely approximate the ideal weight distribution. The program determines the best placement of weights, indicates this to the client, and then moves on to further steps assuming this weight placement was achieved.

- Resistive force model: Once the weight distribution is known, the program must determine the optimal setting of the wheel camber. Changes in wheel camber most directly affect the amount of friction experienced at the road-to-wheel interface, so a resistive force model is needed to incorporate friction. Apart from rolling friction, there are also resistive forces experienced by the car due to aerodynamic drag and bearing resistance. The program models the combined effect of all resistive forces using the experimentally determined "road load" parameter. The resistive force model will be used to inform the conservation model as well as indicate the optimal wheel camber and toe.

- Axle bending model: In order to achieve a given wheel camber, a model is needed to determine how much to bend the axles. The axle bending model will take the weight distribution of the car and the desired wheel camber and toe and determine the deflection to apply to the spindles (at either end of the axle).

2.2 Detailed design

2.2.1 Software

The software tool is a graphical user interface (GUI) compiled in MATLAB's App Designer tool. The tool will provide data entry tables for the client to enter geometry details about the car, hill, and racing conditions. The tool will perform a series of calculations based on the input, and then display resulting plots and figures for the client in an easy-to-use fashion. The output of the program indicates to the client how much weight to place in the soapbox car to make it a certain amount nose heavy or tail heavy, depending on the profile of the hill. The program also indicates how much pre-deflection to add to the spindles to counter-balance the weight of the driver.

2.2.2 Math modeling

The software tool will use two primary math models of the soap box derby car in the background to map all inputs to their respective outputs. The first math model applies force balances to the soap box car as it travels down the hill in order to predict the optimal weight configuration in the car. The second model applies force balance to each axle to determine spindle tip deflections that occur under regular loading. This section will cover each model in detail.

Equation of motion The coordinate reference frame and typical hill profile geometry seen in soapbox derby races is shown in the diagram in figure 9

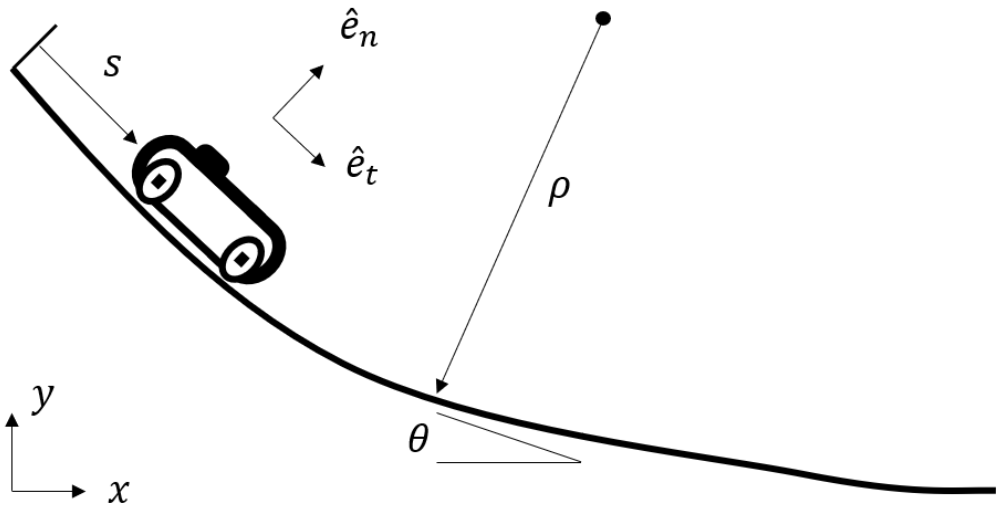


Figure 9: Normal-tangential coordinate reference frame on a typical soapbox derby hill

The hill profile is recorded experimentally using the procedure outlined in appendix B. As a result of the hill profile measurements, the local slope θ and radius of curvature ρ are treated as known values at any s location along the length of the track.

The equations of motion will be developed for the car in the normal ($\hat{e}_n$) and tangential ($\hat{e}_t$) reference frame. The motion of the car will be tracked along the arc length s of the hill, where θ is the local slope and ρ is the local curvature at any location s along the hill. The curvature is calculated in the Cartesian reference frame as

$$\rho(x) = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{\frac{3}{2}}}{\left|\frac{d^2y}{dx^2}\right|} \tag{1}$$

The linear velocity ($\dot{s}$) and acceleration ($\ddot{s}$) of the car in the tangential direction along the track are related to the angular velocity (ω_w) and acceleration (α_w) of the wheels, as shown in figure 10.

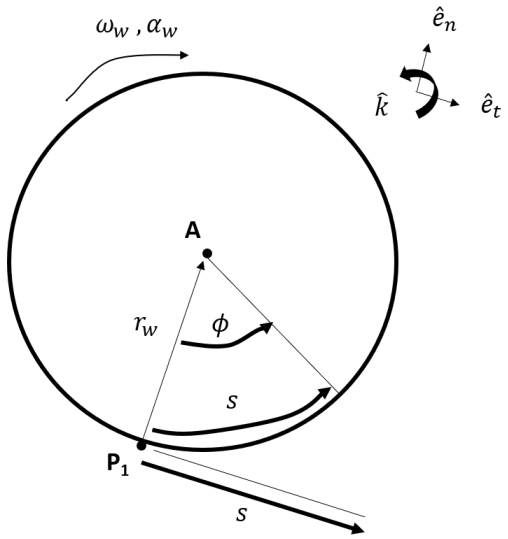


Figure 10: Angular acceleration diagram of the wheel

where r_w is the radius of the wheel. Assuming a no-slip condition at P_1, the linear distance the wheel travels on the track s is related to the angle through which the wheel turns ϕ by

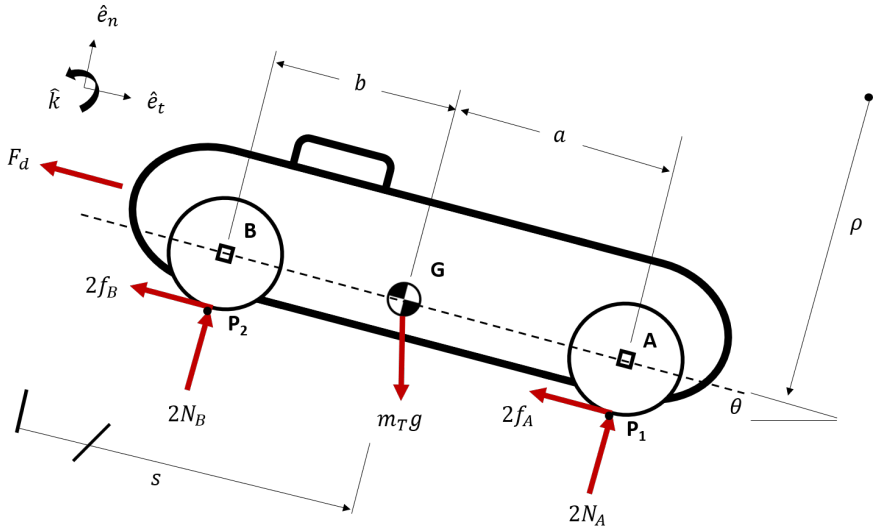
$$s = r_w \phi \tag{2}$$

The first and second time derivatives of s are then given by

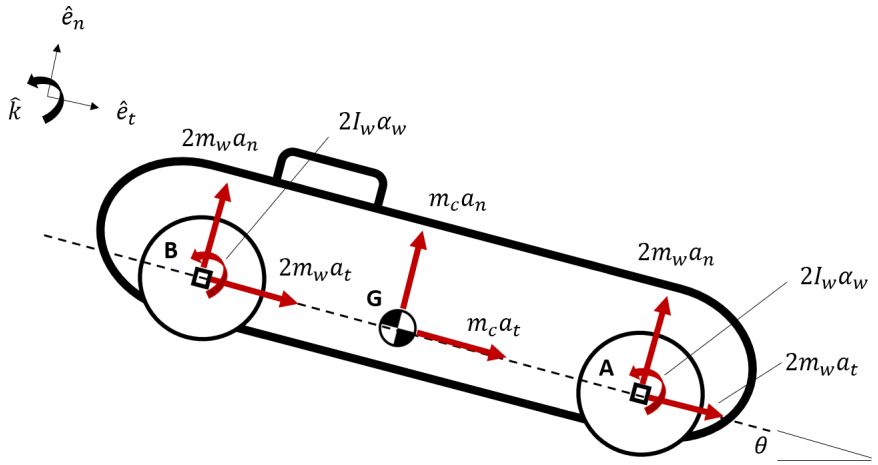
$$\dot{s} = r_w \omega_w$$
$$\ddot{s} = r_w \alpha_w$$

where $\omega_w = \frac{d\phi}{dt}$ is the angular velocity and $\alpha_w = \frac{d^2\phi}{dt}$ is the angular acceleration.

We would like to develop an expression that gives the car's position along the track s over time t. This expression can then be solved to determine which factors most influence the overall race time of the car on any given hill. To develop an expression for s and its derivatives, we apply conservation principles to the system shown in figure 11.



(a) Free-body diagram



(b) Kinetic diagram

Figure 11: Free-body and kinetic diagrams for the soapbox derby car

Conservation of linear momentum applied in the normal direction gives

$$m_T a_n = 2N_B + 2N_A - m_T g cos(\theta) \tag{3}$$

where m_T is the total mass of the car frame, driver, and wheels combined. N_A and N_B are the normal forces on the front and rear wheels, respectively, and a_n is the normal acceleration of the car, as given by

$$a_n = \frac{\dot{s}^2}{\rho} \tag{4}$$

Conservation of linear momentum applied in the tangential direction gives

$$m_T a_t = -2f_B - 2f_A - F_d + m_T g sin(\theta) \tag{5}$$

where f_A and f_B are the friction forces on the front and rear wheels, respectively, and a_t is the

tangential acceleration of the car, as given by

$$a_t = \ddot{s} \tag{6}$$

The drag force F_d is given by

$$F_d = \frac{1}{2}\rho_{air}\dot{s}^2 A_f C_D \tag{7}$$

where A_f is the frontal area of the car and C_D is the drag coefficient. The friction forces are modeled with

$$f_A = \mu_k N_A$$
$$f_B = \mu_k N_B$$

where μ_k is the kinetic friction coefficient. The mass of the car frame and driver (m_c) is related to the total mass and the mass of the wheels by

$$m_T = m_c + 4m_w \tag{8}$$

Conservation of angular momentum applied in the counter-clockwise $\hat{k}$ direction about P_1 gives

$$4I_w\alpha_w - m_T a_t r_w - m_c a_n a - 2m_w a_n(a+b) = m_T g cos(\theta)a - m_T g sin(\theta)r_w - 2N_B(a+b) + F_d r_w \tag{9}$$

where a and b are the moment arms of the front and rear axles, respectively, about the center of gravity. The moment of inertia of the wheels is modeled as

$$I_w = \frac{1}{2}m_w r_w^2 \tag{10}$$

which treats the wheels as small, thin disks. Equations (3), (5), and (9) can be solved simultaneously to obtain the following nonlinear ordinary differential equation (ODE) in terms of s and its derivatives:

$$\ddot{s} + \left(\frac{\rho_{air}A_f C_D}{2m_T} + \frac{\mu_k}{\rho}\right)\dot{s}^2 = g(sin(\theta) - \mu_k cos(\theta)) \tag{11}$$

The ODE can be solved numerically using the Runge-Kutta 4 solver and time-stepped until the car's position s is equal to the track length, at which point the race time is recorded. A list of assumptions used by this model is included here for convenience.

Assumptions

- The axles, center of gravity, and drag force are all aligned horizontally. Any developed moments in the tangential direction about this line are negligible.
- No-slip condition on all four wheels. The friction at each wheel can be reasonably modeled with a single kinetic friction coefficient.
- The car is modeled as a rigid-body. No angular momentum is developed by the car frame in the chosen coordinate system.
- Only 2D dynamics are considered in-plane. The hill profile is assumed constant across the depth of the track in the out-of-plane direction.

- All wheels are identical in mass, moment of inertia, and radius. The wheels are assumed as small, thin disks. Normal and friction forces developed at each wheel are symmetrical about the mid-plane of the car (i.e. N_A is the normal force applied equally at the left wheel and the right wheel on the front axle).

- The acceleration of the wheel centers is equal to the acceleration of the vehicle at the center of gravity.

It is noted here that the center of gravity location does not show up explicitly in Eq. (11). The effect of center of gravity location, x_{cg}, manifests itself in the initial conditions of the numerical ODE solver. The starting location s_0 of the soap box car is set equal to x_{cg}, and the starting velocity $\dot{s}_0$ is always 0. The effect of changing x_{cg} can be determined by moving x_{cg} from the front of the car to the back of the car at several discrete intervals, re-solving the ODE at each interval, and recording the changes in overall race time. In this way, a plot is generated and displayed to the user that shows race time plotted against center of gravity location.

A value that is more useful to the client than center of gravity location is how many pounds to go nose-heavy or tail-heavy. This value is typically measured using two weight scales, one underneath the front axle wheels, and the other underneath the rear axle wheels. Figure 12 shows a free-body diagram of a soapbox car resting level on two weight scales in a typical setup.

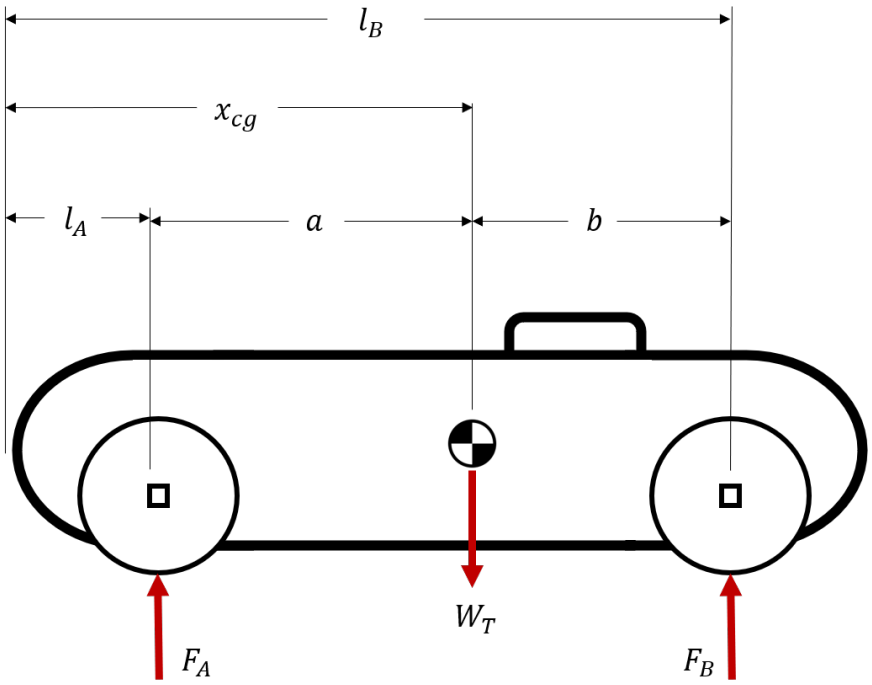


Figure 12: Typical weighing setup of a soapbox car

F_A is the weight reading of the front axle scale (in lbs) and F_B is the weight reading of the rear axle scale. W_T is the overall weight of the car and driver. The distances to the front and rear axles from the front of the car (l_A and l_B respectively) are known geometric values. The geometric

parameters a and b are determined by the placement of the center of gravity x_{cg}, given as:

$$a = x_{cg} - l_A$$
$$b = l_B - x_{cg}$$

The "amount" of weight to go tail-heavy or nose-heavy is defined by Δf:

$$\Delta f = F_B - F_A \tag{12}$$

Positive values of Δf indicate a tail-heavy loading while negative values indicate a nose-heavy loading. A force balance applied in the vertical direction gives:

$$F_A + F_B = W_T \tag{13}$$

Conservation of angular momentum about the center of gravity gives:

$$F_A a = F_B b \tag{14}$$

Solving Eqs. (12), (13), and (14) simultaneously gives:

$$\Delta f = \frac{W_T(a-b)}{(a+b)} \tag{15}$$

Eq. (15) is used to determine how much weight to put either forward or backward in the car based on the optimal center of gravity location determined by the solution of Eq. (11). This value is the final result displayed to the user as a result of this math modeling section.

A simplified version of Eq. (11) is obtained by neglecting drag and friction forces:

$$\ddot{s} = gsin(\theta) \tag{16}$$

which is a simple statement of conservation of linear momentum in the tangent direction. This simplified model is desirable when the friction and drag coefficients are not known or are otherwise hard to obtain. To account for friction, drag, and other resistive forces under this model, a combined "road load" force F_R is applied against the direction of motion:

$$m_T\ddot{s} = m_T gsin(\theta) - F_R \tag{17}$$

The road load force is experimentally determined from the SAE coastdown technique [6] as a quadratic fit to the vehicle's velocity:

$$F_R = a\dot{s}^2 + b\dot{s} + c$$

where a, b, c are the polynomial fit coefficients. The final equation of motion used to evaluate the dynamics of the soap box car is then given by:

$$m_T\ddot{s} + a\dot{s}^2 + b\dot{s} + c = m_T gsin(\theta) \tag{18}$$

Spindle deflection Wheel camber and toe angles are the angles the wheels make with respect to vertically and horizontally aligned datums, as defined in the Nomenclature section. During soapbox derby races, it is desirable to have the camber and toe angles as close to 0 as possible to maintain

proper alignment during the race and reduce axle cross-bind. Camber and toe angles develop as a result of loads placed on the axles that cause deflection at the spindles, where the wheels attach to the axle. To counteract this deflection, spindles are typically pre-deflected before a race in the opposite direction as the expected loading deflection. The math model developed in this section provides an analytical way to calculate the exact amount of pre-race deflection that must be applied to the spindles to counteract the deflection that results from axle loading during a race.

Each axle was modeled as a simply-supported beam with simple supports at the inside edge of each spindle where the bearings from the wheels contact the axle, as shown in figure 13.

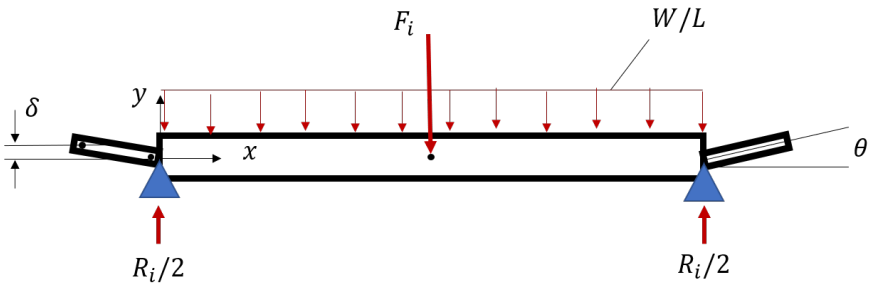


Figure 13: Simply supported beam model of a soapbox car axle.

Each axle is loaded at the center of its length by a single point force F_i and a distributed load $\frac{W}{L}$, where W is the weight of the axle and L is the length of the axle. The axle has a constant square cross-section with an area moment of inertia of $I = \frac{1}{12}t^4$, where t is the thickness of the axle. Two reaction forces develop at the supports such that

$$ma_y = R_i - F_i - W \tag{19}$$

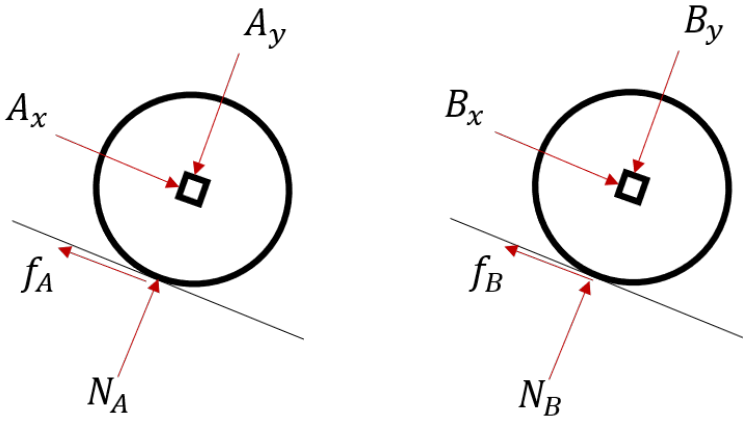
where m is the mass of the axle and a_y is the acceleration of the axle in the y-direction. Assuming small deflections, Euler-Bernoulli beam theory is applied to the system in figure 13 with linear superposition of loading scenarios. The camber/toe angle at both supports ($x = 0$ and $x = L$) is given by:

$$\theta = \frac{F_i L^2}{16EI} + \frac{WL^3}{24EI} \tag{20}$$

where E is the modulus of elasticity. Each axle is assumed to be steel with an average modulus of elasticity of $E = 30.5 \times 10^6$ psi. The spindles are assumed to extend at constant slope outwards from each support. If the length of each spindle is L_s, then the deflection δ at the tip of each spindle is given by:

$$\delta = L_s sin(\theta|_{x=0}) \tag{21}$$

Eqs. (19) - (21) can be applied to both the rear and the front axles in the same manner. It can also be applied for both camber and toe angles by changing the values of F_i, R_i, and a_y accordingly. Figure 14 shows the free-body diagrams for the front and rear axle-wheel assemblies.



(a) Front wheel-axle assembly

(b) Rear wheel-axle assembly

Figure 14: Free-body diagrams of the front and rear wheel-axle assemblies

A_x and A_y are the reaction forces applied by the king pin on the middle of the front axle. N_A is the total normal force on the front axle assembly ($\frac{N_A}{2}$ applied to each wheel), and f_A is the total friction force applied to the assembly. B_x, B_y, N_B, and f_B are the corresponding forces applied to the rear axle assembly. The relationship between the front and rear normal forces is given by conservation of linear momentum applied to the system in figure 12 if it was rotated by an angle θ as the slope of the track incline. Linear momentum in the normal direction gives a modified version of Eq. (13):

$$N_A + N_B - m_T g cos(\theta) = 0 \tag{22}$$

In this case, the normal acceleration is assumed to be negligible because of a large radius of curvature. Solving Eq. (14) and Eq. (22) for N_A and N_B gives:

$$N_B = \frac{m_T g cos(\theta)}{\left(\frac{b}{a}\right) + 1} \tag{23}$$

$$N_A = \left(\frac{b}{a}\right) \frac{m_T g cos(\theta)}{\left(\frac{b}{a}\right) + 1} \tag{24}$$

Again, neglecting the normal acceleration in the systems shown in figure 14, and assuming small axle weight, it can be shown that the vertical reaction forces of the king pin on the axles are equal to the normal forces N_A and N_B. Since the acceleration of the wheels in the tangent direction is non-negligible, conservation of linear momentum on the front axle-wheel assembly gives:

$$ma_t = A_x - f_A \tag{25}$$

The net force A_x - f_A results in a forward acceleration of the axle-wheel assembly ($a_t \neq 0$). The component of A_x that is used to overcome friction f_A is treated as a static force that causes beam deflection in the x-direction. This static force is modeled as

$$f_A = C_{RR} N_A \tag{26}$$

for the front axle-wheel assembly and

$$f_B = C_{RR} N_B \tag{27}$$

for the rear axle-wheel assembly, where $C_{RR} = 0.02$ is an upper-bound estimate of the coefficient of rolling resistance between the ground and the wheels.

The front and rear axle camber and toe angles are determined from Eq. (20), where the point load force F_i is applied in each case as given in table 2.

Table 2: Point load force F_i in each axle beam loading scenario

	Camber	**Toe**
Front axle	N_A	f_A
Rear axle	N_B	f_B

2.2.3 Tachometer

The resistive force model is supplied experimental velocity data from a custom tachometer. The tachometer functions by means of an infrared sensor, which shoots a beam of infrared light at the wheel while the wheel spins. All sensor specifications are included in appendix C. The wheel itself is black and absorbs the infrared light. The wheel has 8 equidistant strips of white tape on it, which reflect infrared light back to the sensor when they cross the beam's path. The sensor records a blip every time a beam of infrared light is reflected back to it. These blips are registered on an Arduino Nano, which records the time it takes for the sensor to register 24 blips (3 revolutions), and divides the 3 revolutions by that time to get an angular velocity value in revolutions per second. The Arduino also keeps track of how long it has been recording data since it was turned on. Each time a velocity data point has been calculated, the Arduino records the value and the time stamp of that value onto a micro SD card for post processing. A diagram of the custom tachometer attached to the soapbox derby car axle is shown in figure 15.

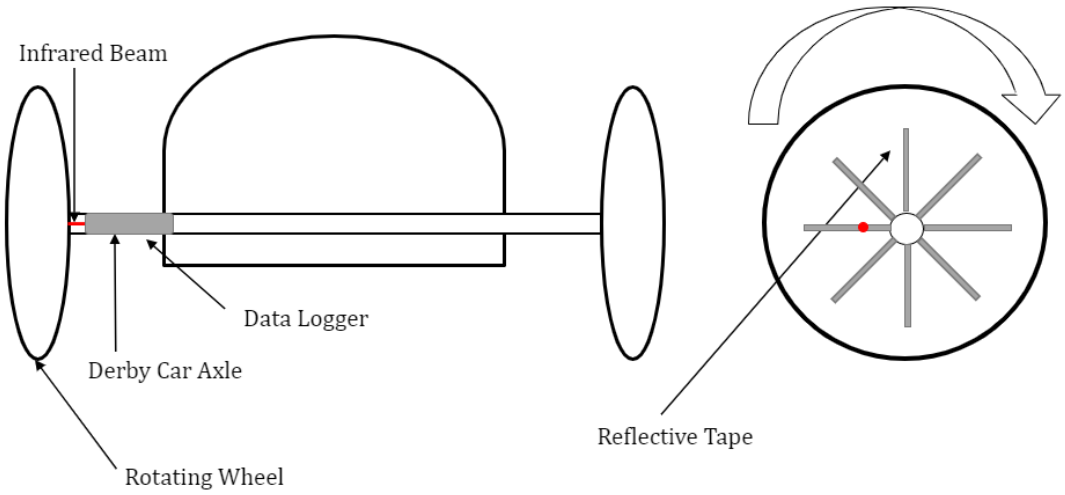


Figure 15: Functional diagram of custom tachometer data logger

The tachometer is contained within a 3D printed housing that is mounted on the axle of the soapbox

car in place of one of the car's standard aerodynamic fins. The device operates under the power of a 9V battery mounted inside the housing next to the Arduino, and is turned on and off by means of a toggle switch. The switch is connected to a long enough wire to be operated by the driver while the vehicle is in motion. A wiring diagram for the tachometer is included in appenidx D. On occasion, the infrared sensor will need to be re-calibrated to adjust for changing environmental ambient light. The calibration procedure is included in appendix E.

3 Prototype

3.1 Software

A prototype of the final software tool was created in MATLAB's App Designer toolbox. The user is first presented with a home screen that directs them to an input data page, as shown in the graphical user interface (GUI) in figure 16

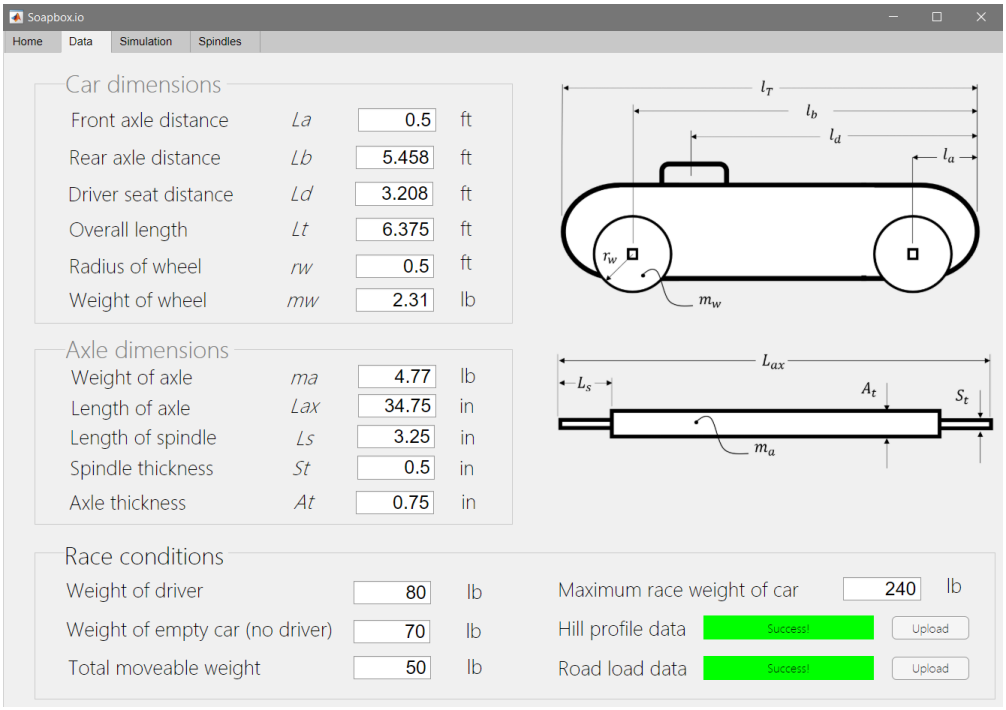


Figure 16: Data input page in the Soapbox.io software tool

On this page, the user inputs details about the geometry of the car and axle, including the overall length, the radius and mass of the wheels, the weight of the driver and car, and the location of the driver's seat. The user is provided a clickable button that allows them to upload an Excel file that contains data about the profile of the race track hill. This hill profile data is collected according to the procedure outlined in appendix B. Another button is provided that allows the user to optionally upload road load velocity data to improve the fidelity of the math models. This velocity data comes directly from the custom tachometer as detailed in section 2.2.3. The next page of the software is the simulation tab, as shown in figure 17.

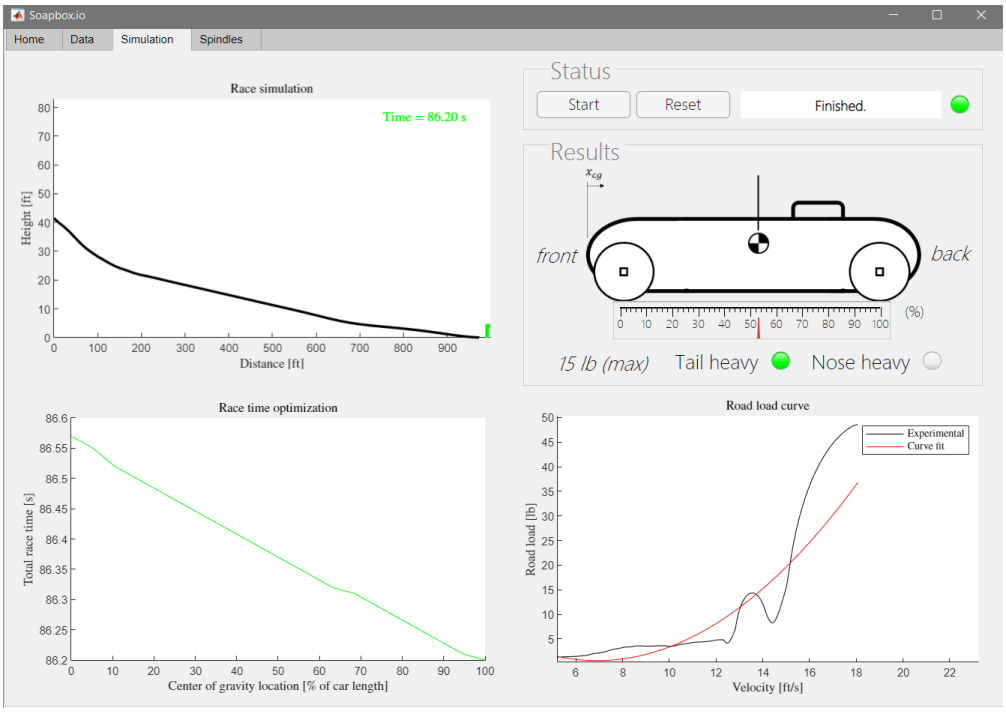


Figure 17: Simulation page in the Soapbox.io software tool

On this page, the user can start the simulation. The software will pull the data input on the previous page and solve the equations presented in section 2.2.2. The software will iterate over several weight distributions to find the one that minimizes the overall race time. The GUI has a status bar that indicates the progress of the calculations. Once the calculations are complete, the GUI will plot the hill profile and weight distribution results on the left side of the simulation tab. In the results panel, the GUI will indicate where the ideal center of gravity should be located, and how much weight to place either forward or backward to make the car nose heavy or tail heavy. If road load data was optionally loaded on the data page, then the road load quadratic fit curve will be displayed in the bottom right of the simulation page.

Figure 18 shows a screenshot of the spindle-bending page.

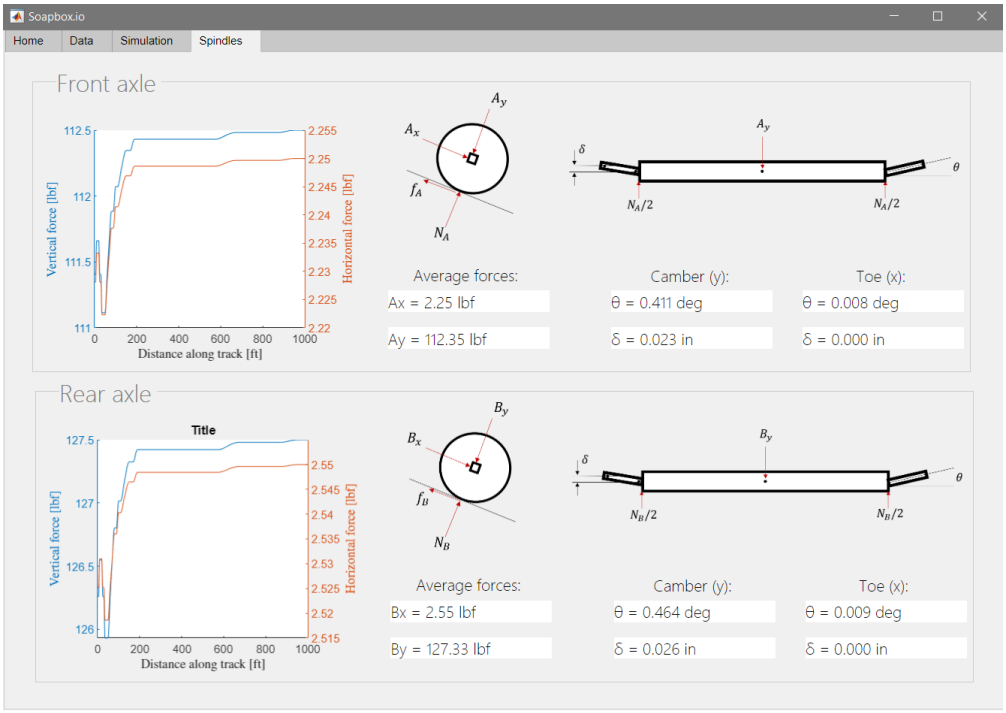
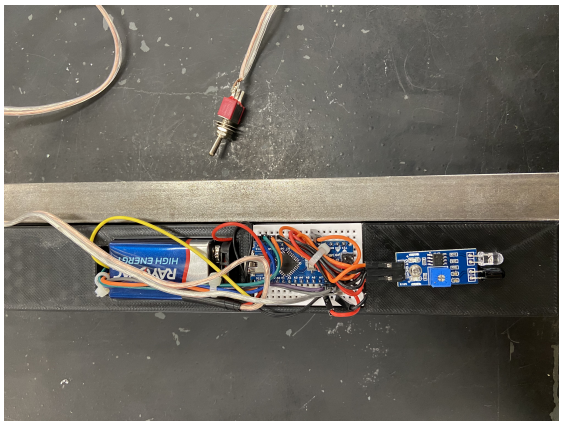


Figure 18: Spindle-bending page in the Soapbox.io software tool

The spindles page contains two results panels displaying camber, toe, and deflection results for both the front and rear axles. A plot of the reaction forces on each axle over the length of the track is provided, as well as the average forces in both the normal and tangential directions. These forces are used directly to calculate the expected camber and toe angles based on the current loading conditions, as outlined in section 2.2.2. The recommended pre-race spindle deflection values are also displayed in each of the results panels (in inches). Camber deflection values are typically on the order of 0.020 in, and toe deflection values are typically negligible due to the low estimated static friction forces acting in the tangential direction.

3.2 Tachometer

The Tachometer prototype matches the general design outlined in the section 2.2.3, and can be activated and deactivated by the driver by means of a toggle switch. The SD card module is mounted on the opposite side of the housing from the rest of the electronics in order to make access of the SD card easier when the logger is attached to the car. The 3D printed housing consists of two parts: a main housing and a shield. The main housing has slots to place the battery and the breadboard holding the Arduino. There are also screw holes for mounting the housing to the axle, as well as smaller screw holes to secure the SD card and IR sensor modules. The shield is used to protect the the electronics from weather conditions, as well as minimize the ambient sunlight able to reach the IR sensor, as the sensor is very sensitive to ambient light. The housing and housing shield fit together with their screw holes lined up before being attached to the axle in place of one of the car's aerodynamic fins. The final assembled prototype is shown in figure 19.



(a) Tachometer assembled to axle (no shield)

(b) Tachometer assembled with shield cover on

Figure 19: Tachometer prototype

The full bill of materials for this prototype is included in appendix F. The CAD drawings for the housing and the housing shield cover custom parts are included in appendix G. The previous prototype for this project is also included in appendix H for documentation purposes.

4 Design verification

This section will evaluate how well the client needs have been met by the design solution, as well as provide validation for each sub-process of the overall solution. Table 3 shows how the final design solution ranked against our technical specifications, with a score of 1 being the worst and a score of 3 being the best. Each ranking was weighted by the weight of each individual need.

Table 3: Ranked performance of the design solution against the technical specifications

			Technical specifications					
No.	Client needs	Weight	**Δ Race time (s)**	**User survey**	**Calculation time (min)**	**Collection time (min)**	**Scalable**	**Free**
	(target)		> 0.01	8/10	< 1	< 60	Yes	Yes
	(actual)		0.05	n/a	0.5	45	Yes	Yes
	Met?		Yes	n/a	Yes	Yes	Yes	Yes
1	Feasible tuning settings	5	3					
2	User-friendly	4		(-)	3	3		
3	Quick processing time	3			2			
4	Easy data collection	3		(-)		2		
5	Easily scalable	2	2				1	
6	Free software	1						3
Total score: 60 Target: 67		Sum:	19	(-)	18	18	2	3

The resulting score on this design solution was a total of 60 out of the target 67. Due to timing constraints at the time of writing of this document, a user survey has not yet been given to our client, which we believe will provide the additional points to reach our final target with a relative margin of safety. Initial simulations of soapbox derby races on Wilbur Shaw hill indicate that moving weight back in the car by the maximum allowable amount of 15 lbs will provide an improvement in race time of 0.05 seconds, which exceeds the race time technical specification and earns a ranking of 3 out of 3. It is noted that further improvements are also possible by applying the program-recommended spindle deflections, but a full statistical study of this would be necessary to fully justify the effect on race time improvement.

The calculation time was measured as an average of 30 seconds, depending on the machine running the program. This earns a 3 out of 3 and meets the calculation time specification. The data collection time on Wilbur Shaw hill was recorded as 45 minutes, which falls under the target 60 minutes that the client has available before a race. The Matlab software has built-in *if* statements to check for the presence of road load data, which helps improve the fidelity of the model when included, but is not necessary to run the program. This functionality allows us to say we have met our scalability criteria, but with a low score of 1 out of 3, because further changes are needed to account for even higher fidelity models. The Matlab program has been tested to compile and run on a computer that does not have an expensive Matlab software license installed, so the free software specification has been met in full.

4.1 Soapbox race simulation

In order to predict the optimal weight configuration in a soapbox car based on the weight of the driver and the profile of the race track, the Runge-Kutta 4 numerical solver was used to solve the ODE Eq. (18) under the assumptions listed in section 2.2.2. To check the fidelity of the model and the convergence of the time-stepper, race times were gathered from two separate sources and compared to the predicted race times from the numerical solver. The Wilbur Shaw derby hill was modeled in Solidworks and a motion analysis was performed assuming no friction between the tires and the track. The numerical solver was also set to use a frictionless model for comparison. Then, experimental race time data was collected at Wilbur Shaw hill in person under the same loading conditions as was used in each of the simulations. Three trials were taken for each setup and averaged, as summarized in table 4.

Table 4: Race time prediction results

Runge-Kutta solver	**Solidworks motion**	**Experimental**	**Percent difference**
27.9 s	28.5 s		2.1%
27.9 s		28.8	3%

The 3% difference between the Runge-Kutta solver and the experimental data is most attributable to the presence of frictional losses in the experimental test, which would cause the numerical prediction to be lower than the real experimental values.

In addition to the agreement between the numerical solvers and the experimental data presented in table 4, the model used in our Runge-Kutta Matlab solver predicts better performance on concave-shaped tracks when weight is placed tail-heavy in the car, and it predicts better performance on convex-shaped tracks when weight is placed nose-heavy in the car. This behavior is apparent not only intuitively, but also in prior soapbox physics modeling work [3]–[5]. It is the opinion of the authors that most racing conditions will call for either a completely front-loaded or a completely tail-loaded car as the optimal weight distribution, rather than some intermediate loading setup. However, the benefit of the current approach is that if some race track and driver combination is conceived that would cause an optimal loading to be at an intermediate weight distribution rather than at either extreme, then the software will detect and present the corresponding results to the user in a practical and easy-to-use fashion.

4.2 Velocity data collection

The velocity data collected using the data logger is used to model the mechanical losses and resistive forces in a soapbox car. The model for these forces is the SAE road load polynomial:

$$F_R = a + bv + cv^2 \tag{28}$$

which approximates the forces slowing a vehicle down as a quadratic function of the vehicle's velocity. This has the advantage of not forcing the model to account for individual losses (air drag, friction, etc.). Instead, it approximates the sum of all forces slowing the car in one combined term. In the above equation, F_R represents the combined resistive force on the vehicle, v is the car's velocity, and the coefficients a, b, and c are polynomial fit constants that vary based on the car's setup. The data logger collects the car's wheel speed in revolutions per second, as well as the time

stamp in milliseconds that each velocity data point is taken. The velocity data collection procedure is outline in appendix I. The wheel speed is converted into overall vehicle speed assuming no wheel slippage, using:

$$v = r_w \omega \tag{29}$$

where v is the vehicle speed in ft/s, r_w is the radius of the wheel in ft, and ω is the measured angular velocity in rad/s. From conservation of linear momentum on a car coasting down on a flat track:

$$ma_t = -F_R \tag{30}$$

Where m is the total mass of the car, and $a_t = \frac{dv}{dt}$ is the acceleration of the car. The vehicle's speed is numerically differentiated and multiplied by the car's mass m to get a curve of the force F_R with respect to time. The force values are then graphed against the vehicle's velocity to get the road load curve (F_R [lb] v. v [ft/s]). Finally, a quadratic polynomial fit is applied to the road load curve to get the constant fit parameters a, b, and c. An example road load curve fit to lab setting experimental data is shown in figure 20. The raw experimental data is included in appendix J.

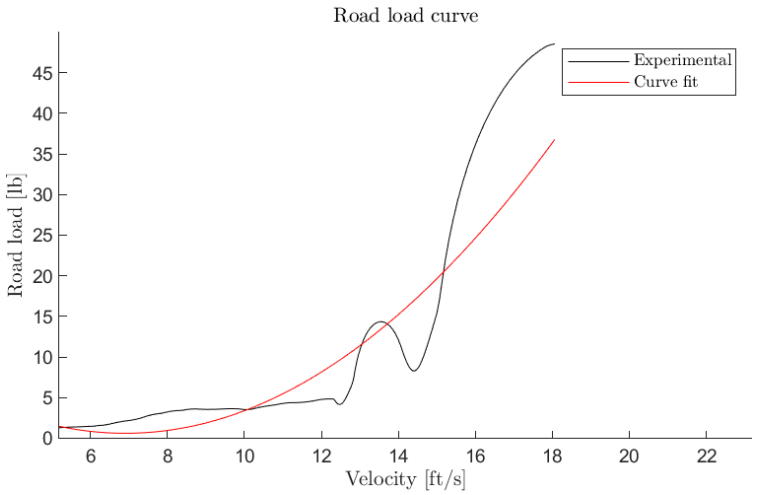


Figure 20: An example experimentally-determined road load curve

In principle, if any adjustment to the car reduces the road load on the vehicle, as determined by the data logger road load curve, then there will be fewer mechanical losses slowing the car down, resulting in a faster race time.

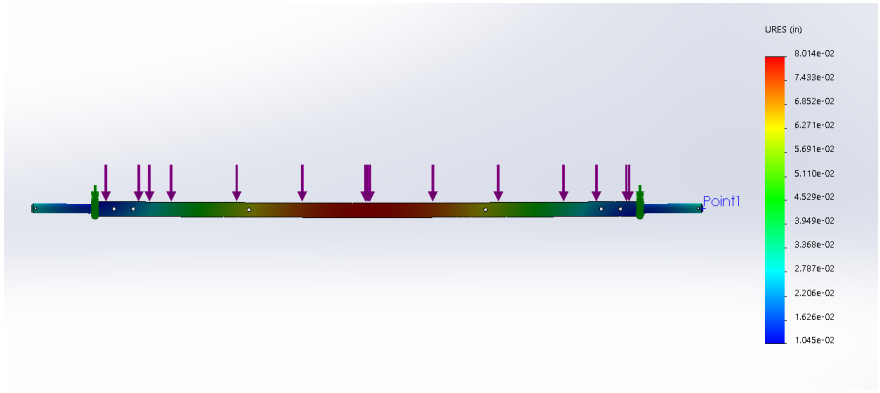
In order to validate the accuracy of the tachometer, we compared it to a commercially available handheld tachometer by having both devices measure the angular speed of a wheel at the same time and comparing the results. During this test, the commercial tachometer and our custom tachometer yielded a roughly 2.5% difference from each other, as shown in table 5. The specifications for the handheld tachometer are included in appendix C.

Table 5: Tachometer accuracy verification

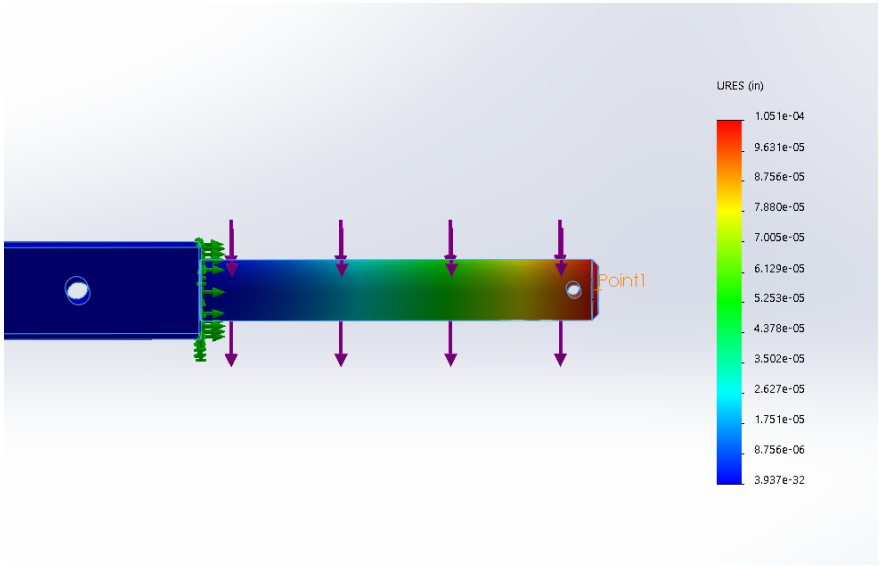
Commercial tachometer	**Prototype**	**Percent Difference**
177.5 rpm	181.8 rpm	2.42%

4.3 Axle spindle deflection

By splitting up the deflection into normal (camber) and tangential (toe) directions, we can model the forces acting upon each axle and in each direction during a race, as detailed in section 2.2.2. Each axle was modeled as a simply supported beam, and the end slope was calculated. This was used to geometrically determine the deflection the spindle would undergo if it started level before being loaded on the car. The simply-supported beam model was validated against a Solidworks finite element analysis (FEA) of the axle loading, for both camber and toe deflections as shown in figure 21.



(a) Camber deflection model



(b) Toe deflection model

Figure 21: Solidworks FEA axle bending models

The concentrated force at the center of the axle for the beam bending calculations was set equal to the average force seen by the car going down the hill. Table 6 shows the comparison of our numerical results to the Solidworks FEA study for the rear axle loading and deflection.

Table 6: Rear axle bending results

	Beam-bending model	**Solidworks FEA**	**Percent difference**
Camber deflection (in)	0.026	0.028	7.7%
Toe deflection (in)	0.000	0.000	n/a

The 7.7% difference between our model and the Solidworks FEA is most attributable to the difference in how the spindles are treated. In the beam bending model described in section 2.2.2, the spindles were treated as straight, linear deflections rising away from the simple supports. The full FEA analysis in Solidworks however, treats the spindles as under an applied load, with a non-constant slope. This would cause the FEA study to predict greater deflections at the tips of the spindles due to the higher curvature. Overall, this study validates our simplified beam-bending model that is incorporated into the software tool for spindle deflection prediction. The fidelity of the model could be improved by replacing the linear spindle assumption and allowing the spindles to bend with the rest of the axle before calculating tip deflections.

Since the resolution of the spindle deflection measurement device is limited to 0.001 inches, the predicted toe deflections by both the beam-bending model and the FEA study are negligible for this case; a percent difference calculation is not provided. In general, since the car is accelerating in the tangent direction down the track, only the component of the forward-acting force used to overcome friction is treated as a static force that causes beam-bending in the axle. Because the friction force is much smaller than the gravity force pulling the car down the hill, the beam deflections seen in the tangent direction (toe) are much smaller than the deflections seen in the normal (camber) direction.

5 Lessons Learned

- Troubleshooting and real-world testing bring to light a lot of issues that do not come to light otherwise, so testing early and looking for unexpected problems will pay off later.
- Rethinking the concept of the design can slow down meetings and lower productivity, especially late in the project. As such, hard deadlines need to be set about when certain aspects of a project will make it into the final design.
- While compartmentalizing tasks is important, it is also essential that all members of the team have a working understanding of all parts of the project, so as to avoid confusion about aspects that a member may not have directly worked on.

References

[1] "Super stock car plans," *Soap Box Derby*, Aug. 2017.

[2] (2020). "Indianapolis soap box derby," [Online]. Available: `https://www.soapboxderby.org/indianapolis.aspx` (visited on 11/04/2020).

[3] P. Gale, "Physics involved in soap box derby racing," *The Marietta (GA) Soap Box Derby*, 2008. [Online]. Available: `http://www.midmosbd.org/sbd_physics.pdf`.

[4] B. Mann, M. Gibbs, and S. Sah, "Dynamics of a gravity car race with application to the pinewood derby," *Mechanical Sciences*, vol. 3, no. 2, pp. 73–84, 2012.

[5] H. Driscoll, A. Bullas, C. King, T. Senior, S. Haake, and J. Hart, "Application of newtonian physics to predict the speed of a gravity racer," *Physics Education*, vol. 51, no. 4, 2016.

[6] C. Chapin, "Road load measurement and dynamometer simulation using coastdown techniques," *SAE Transactions*, pp. 2491–2505, 1981.

Appendices

A Interview notes

- **Interviewer:** Joshua Eckels
- **Notetaker:** Shane Saylor, Charles Earle
- **Observer:** Pascal Liberge
- **Interviewee:** Bob Getts
- **Date:** October 4, 2020
- **Time:** 2:00PM - 4:00PM EST
- **Location:** Wilbur Shaw Soap Box Derby Hill, 2200 W 30th St, Indianapolis, IN 46222

Interview Questions.

1. **What is your background and experience?**

   - Rose-Hulman graduate, class of 1991. Worked at Johnson Controls. Now in Environmental Controls Company, HVAC and airflow management. Director of Indiana soap box derby for 4 years.

2. **What inspired your role in this project?**

   - Interest in soap box since he was a kid, both of his sons participate in soap box derby, he's the director of the Indianapolis team and the races at Wilbur Shaw hill.

3. **What is the context for the project?**

   - Wants to grow the STEM school competition in the area, improve team's performance at the championships in Akron, OH

4. **Is there a timeline for the project?**

   - Big competitions in May and June, but races just about once a month, even indoors on occasion. Just tested and completed by the end of school year, can be continued after we graduate.

5. **What is the budget for the project?**

   - Bob will pay for things as we go; within reason

6. **What specific purpose will the project be used for?**

   - Optimize soap box tuning settings to improve race time. They typically do all the tunings experimentally. They would like to know if there are optimal values and how you can find them mathematically.

7. **What major functions should the product provide?**

- It should input data about the hill, the car, the driver etc. and spit out optimal suspension settings, such as torque on king pin and axles, front/tail heavy weight. Eliminate cross-bind.

8. **Who will use this product?**

   - Typically adults tune the suspension for the kids using spindle binding tools and special gauges

9. **Where will the product be used? Where and how can it be tested?**

   - They have accurate timing equipment at Wilbur Shaw, can test the car there. Used on various soap box derby hills, always paved, different profiles.

10. **Are there any rules and restrictions we should consider?**

    - There are rules on the hardware. They are kit cars basically. Not a lot of customization so that the races come down to how well the kids drive. Level playing field. Posted rules online for soap box. They can adjust various things like camber, toe, cross-bind, and weight. Car body is just wood floor with axles and spindles. Wheels are clipped on (several pictures taken onsite as well)

11. **Are you aware of anyone that has tried this before?**

    - Not that we are aware. There are "winning ingredients" books out there that give tips and tricks on how to slightly improve time, just sales people

12. **What are typical race times that you see in your team, what would be a good improvement?**

    - Races typically come down to tenths of a second. If you're ahead by more than that then you're doing pretty good. We have really accurate timing equipment here (0.001) resolution.

13. **How do you plan on using this product on a typical race day?**

    - Spend an hour roughly before the race walking the track and doing some last minute tuning. We have to weigh the cars and adjust the weights as well. Typically there are two runs per car where you switch wheels with the opponent. Can do small tuning in between each race but only a couple minutes.

B Hill profile data collection procedure

Race time is very dependent on the shape of the track from the starting line to the finish line. The software program must be given information on the shape of the race track to correctly predict race times and optimize weight placement in the car. On the data page of the software tool, there is a clickable button for uploading an Excel (.xlsx) file that contains information about the shape of the race track. The data in that file is formatted in two columns. The first column is the distance s one has walked from the starting line, where the first data point is always $s = 0$ right at the starting line. The second column is the slope of the hill (in degrees above horizontal). A series of (distance, slope) data points are recorded in these two columns by walking the length of the track from beginning to end. A distance measuring wheel, such as the one shown in figure 22, is used to measure the distance that has been walked along the length of the track.

Figure 22: Distance measuring wheel

The slope of the track at a point is measured with an inclinometer. Most smart phones come packaged with a built-in inclinometer that gives slope readings in degrees, such as shown in figure 23.

Figure 23: Smart phone inclinometer

The data collection procedure is as follows:

1. Reset the distance wheel to a reading of 0 ft. Ensure the inclinometer reads 0 degrees when placed on a flat, level surface.

2. Place the distance wheel at the starting line of the track. If a ramp is present, place the distance wheel at the base of the ramp.

3. Place the inclinometer at the starting line and record the slope readout of the sensor (in degrees). If a ramp is present, record the slope of the ramp instead. Record a data point as the current slope readout at a distance of 0 ft. Format the data so that distance (ft) is in the first column of an Excel sheet and slope (degrees) is in the second column. A downhill slope should be recorded as a negative value, and an uphill slope should be recorded as a positive slope.

4. Walk with the distance wheel rolling on the ground until the distance wheel reads about 5 ft. Record another inclinometer slope reading as performed previously.

5. Continue walking the length of the track in this fashion. When the track appears to change incline rapidly, decrease the distance between each measured data point. The opposite is true; if the track appears to maintain a constant slope, increase the distance between each measurement until the slope readings start changing again.

6. Finish the data collection procedure when the distance wheel reaches the finish line.

7. Save the data as an Excel file in the two-column format previously described. Upload this file to the software tool directly as is when prompted.

C Sensor specifications

Ardunio IR sensor

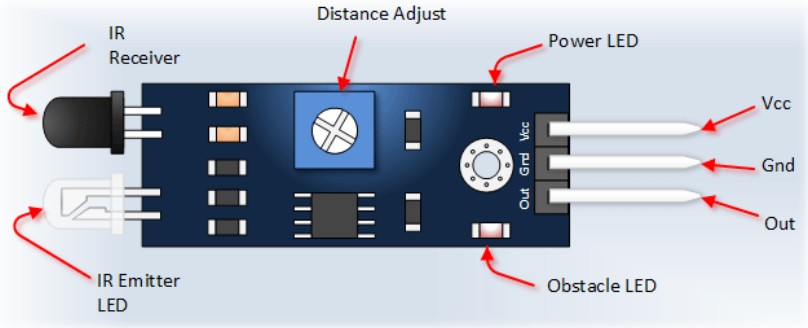
Arduino Infrared Collision Avoidance

This is yet another one of those modules with cool possibilities. You could for example, sound an alarm when something got too close or you could change the direction of a robot or vehicle.

The device consists of an Infrared Transmitter, an Infrared Detector, and support circuitry. It only requires three connections. When it detects an obstacle within range it will send an output low.

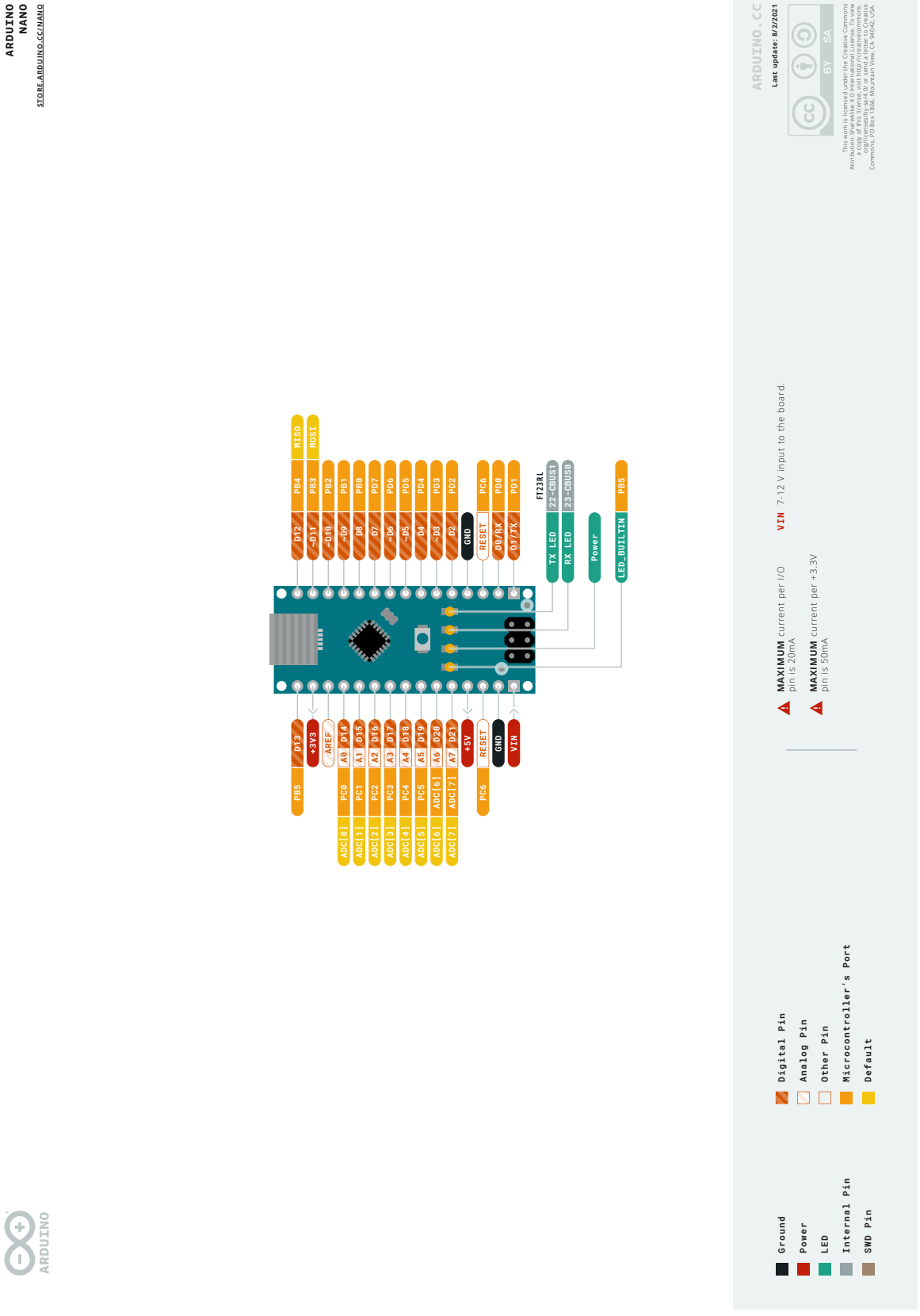
IR Obstacle Detection Module Pin Outs

The drawing and table below identify the function of module pin outs, controls and indicators.



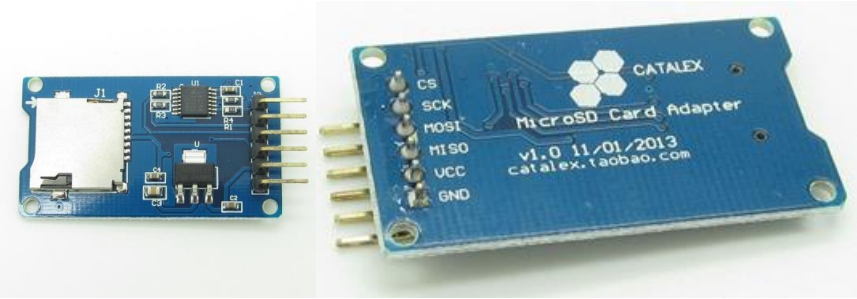
Pin, Control Indicator	Description
Vcc	3.3 to 5 Vdc Supply Input
Gnd	Ground Input
Out	Output that goes low when obstacle is in range
Power LED	Illuminates when power is applied
Obstacle LED	Illuminates when obstacle is detected
Distance Adjust	Adjust detection distance. CCW decreases distance. CW increases distance.
IR Emitter	Infrared emitter LED
IR Receiver	Infrared receiver that receives signal transmitted by Infrared emitter.

Arduino Nano microcontroller



eBay Search:

Micro SD Card Micro SDHC Mini TF Card Adapter Reader Module for Arduino



Description

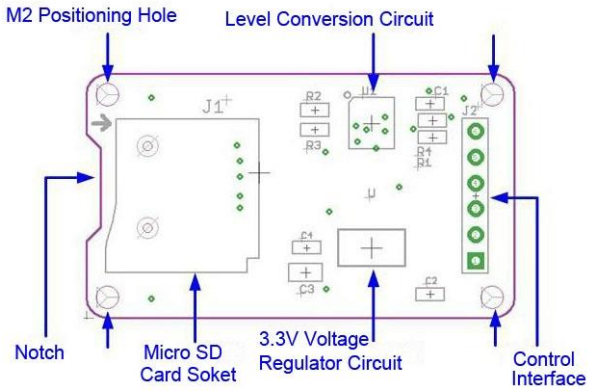
- The module (MicroSD Card Adapter) is a Micro SD card reader module for reading and writing through the file system and the SPI interface driver, SCM system can be completed within a file MicroSD card
- Support Micro SD Card, Micro SDHC card (high speed card)
- Level conversion circuit board that can interface level is 5V or 3.3V
- Power supply is 4.5V ~ 5.5V, 3.3V voltage regulator circuit board
- Communications interface is a standard SPI interface
- 4 M2 screws positioning holes for easy installation
- Control Interface: A total of six pins (GND, VCC, MISO, MOSI, SCK, CS), GND to ground, VCC is the power supply, MISO, MOSI, SCK for SPI bus, CS is the chip select signal pin;
- 3.3V regulator circuit: LDO regulator output 3.3V for level conversion chip, Micro SD card supply;
- Level conversion circuit: Micro SD card to signal the direction of converts 3.3V, MicroSD card interface to control the direction of the MISO signal is also converted to 3.3V, general AVR microcontroller systems can read the signal;
- Micro SD card connector: self bomb deck, easy card insertion.
- Positioning holes: 4 M2 screws positioning holes with a diameter of 2.2mm, so the module is easy to install positioning, to achieve inter-module combination.

SD card reader

Interface Parameters:

Items	Min	Typical	Max	Unit
Power Voltage VCC	4.5	**5**	5.5	V
Current	0.2	**80**	200	mA
Interface Electrical Potential	3.3 or 5			V
Support Card Type	Micro SD Card(<=2G), Mirco SDHC Card(<=32G)			—
Size	42X24X12			mm
Weight	5			g

Mirco SD Card Interface Module:



Commercial Handheld Tachometer

EXTECH INSTRUMENTS
A FLIR COMPANY

Experience the Extech Advantage

PRODUCT DATASHEET

Mini Laser Photo Tachometer Counter

Laser Light Source
Used for greater accuracy and longer measuring distances taking non-contact RPM measurements

Features:

- Make non-contact RPM measurements of rotating objects
- Use reflective tape on object to be measured and point the integral laser
- Memory button holds last reading and recalls min/max readings
- Counter function counts up to 99,999 revolutions
- Easy to read large 5 digit LCD display
- Rugged double molded housing for better grip
- Complete with 9V battery and reflective tape

Non-contact fan blade RPM or REV testing

Specifications	
RPM range	2 to 99,999rpm
Count range	1 to 99,999rev (revolution)
Max Target Distance	1.6ft (500mm)
Basic Accuracy	±0.05%
Resolution	0.1rpm, 1rev
Memory	Min/Max/Last
Dimensions	6.2x2.3x1.6" (160x60x42mm)
Weight	5.3oz (151g)

Ordering Information:

461920.....Mini Laser Photo Tachometer Counter
461920-NIST.....461920 with NIST Certificate
461937.....Spare reflective tape (23" each strip), 10pk

CE 1 YEAR WARRANTY

www.extech.com

D Wiring diagram

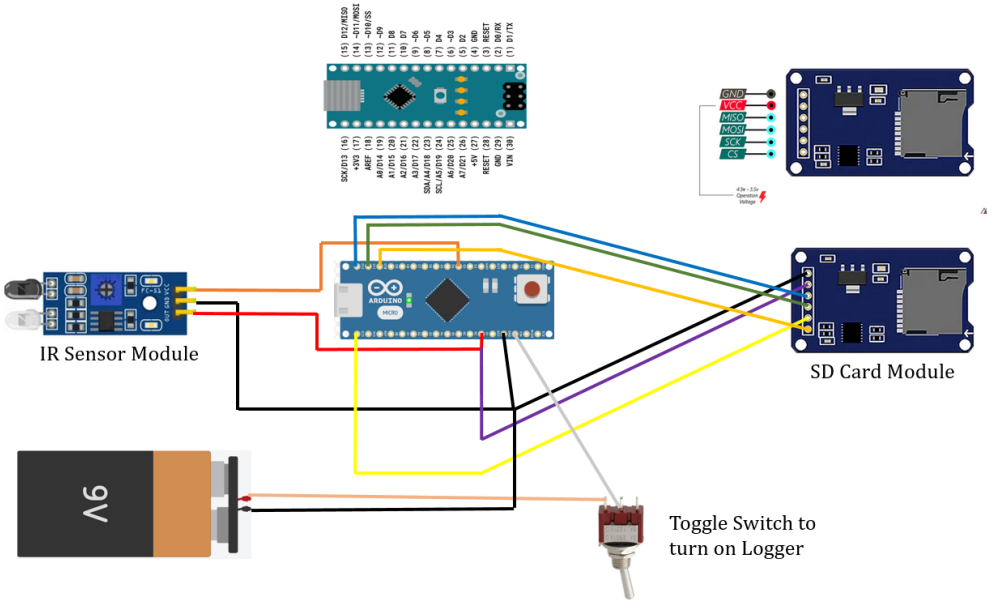


Figure 24: Wiring Diagram for Data Logger

Figure 24 shows the how the wiring of the tachometer is configured, with the wires color coded to match those on the actual prototype. This diagram is here to help diagnose issues or aid in rewiring should any of the leads come undone. The leads going to the toggle switch are not to scale, and on the actual prototype are much longer so that the switch can be reached easily by the driver.

E IR sensor calibration

To calibrate the infrared sensor, place the axle on a flat surface such as a desk, with the data logger attached without the shield. Attach a calibration wheel to the end of the axle, with the white tape facing inwards towards the IR sensor. A calibration wheel has 8 same-size strips of white tape spaced evenly radially about the center of the wheel, as shown in figure 25.

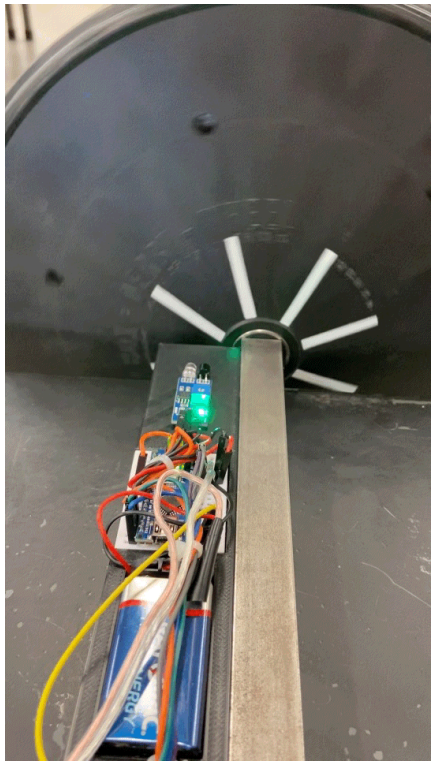


Figure 25: Tachometer and wheel in place during calibration

The end of the axle with the wheel should be hanging off of the desk such that the wheel can spin freely. This should be done indoors, to prevent any sunlight from interfering with the sensor. Begin by turning the potentiometer (small blue block with a Phillips screw head location, located behind the IR LED bulbs) on the sensor counterclockwise as far as it will go. This means that the sensor is set to its least sensitive setting. When the device is on, only one of the green LEDs on the sensor should be active (the power indication light). Spin the wheel and gradually turn the potentiometer clockwise with a Phillips screwdriver to increase sensitivity. When the second green LED flashes every time a piece of white tape goes by, but is not triggered by the black surface of the wheel, then the sensor should be properly calibrated. If the sensor still needs fine tuning outdoors, there is a small hole in the shield that allows for screwdriver access to the potientiometer.

F Bill of materials

The components used to create the tachometer are shown in figure 26 and listed, along with their prices, in table 7. The items listed and shown are what was chosen based on shipping time, price and practicality. Different devices performing the same tasks may be used to accomplish the same functions.

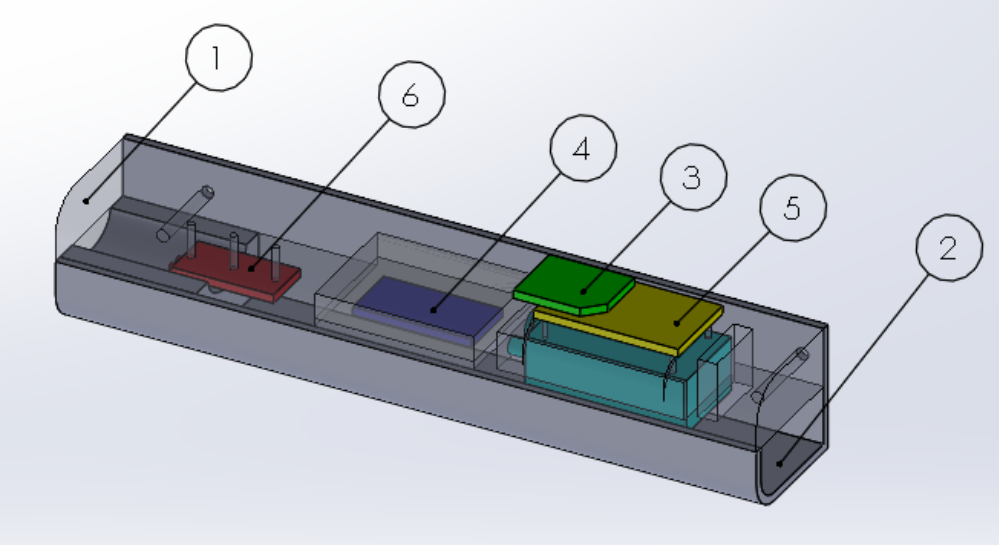


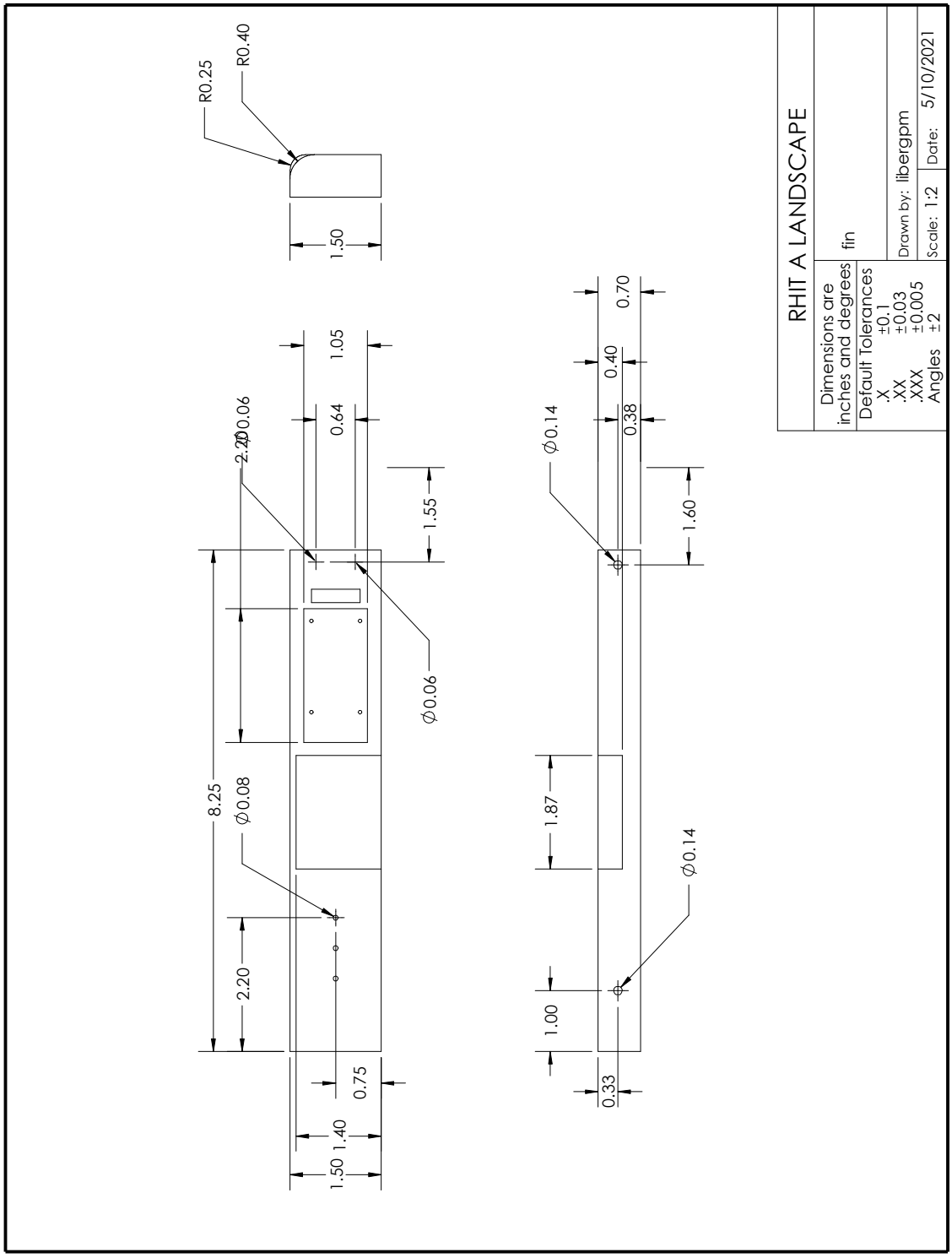
Figure 26: Assembly of all the electronics and housing

Table 7: Bill of Materials

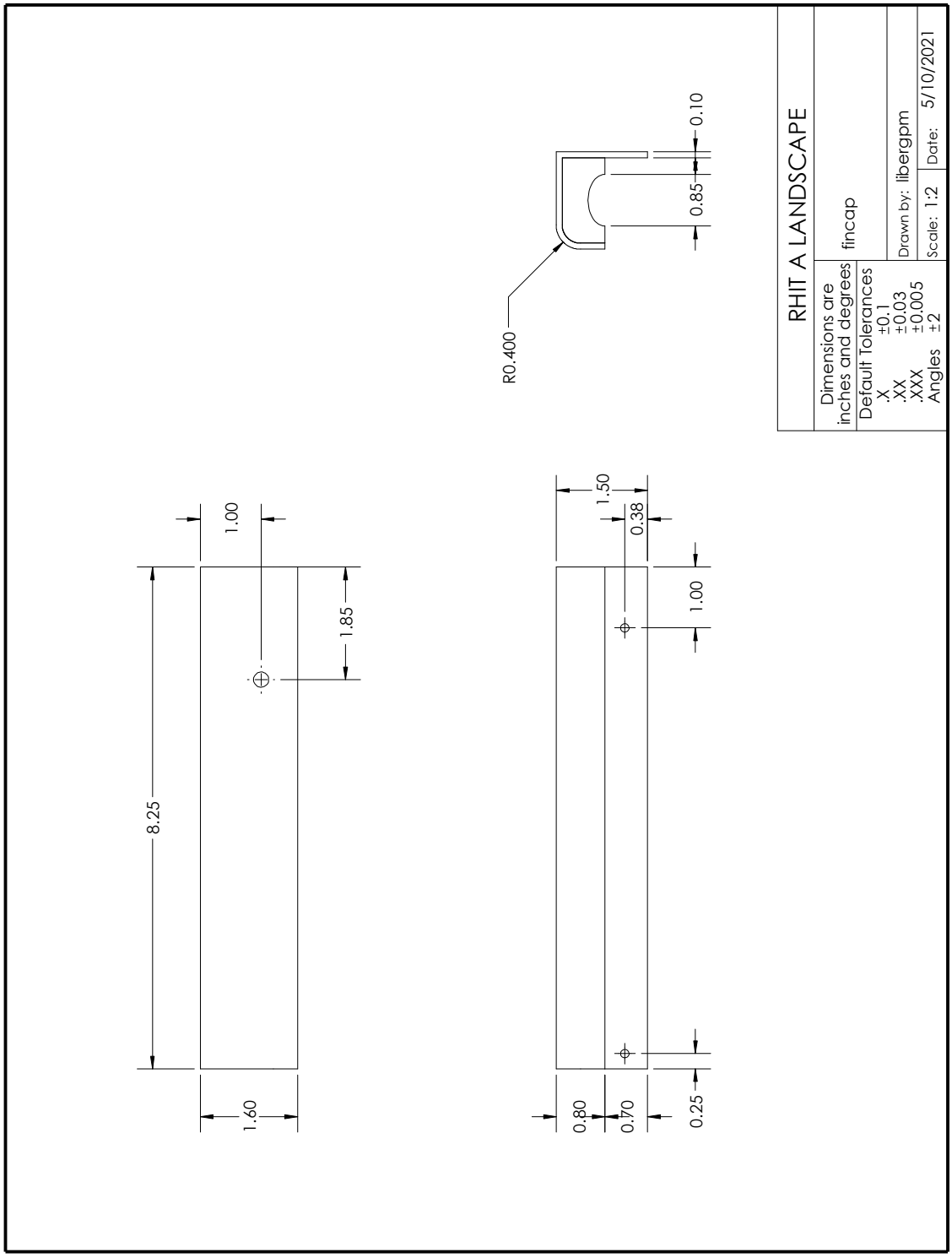
Number	Part	Description	Unit Cost	Quantity	Price
1	Housing	3D Printed from custom model	$0.00	1	$0.00
2	Housing Shield	3D Printed from custom model	$0.00	1	$0.00
3	SD Card	Stores time and velocity data	$5.74	1	$5.74
4	Arduino Nano	Microcontroller for Tachometer	$9.99	1	$9.99
5	Arduino SD Card Module	Allows data to be stored on SD Card	$5.99	1	$5.99
6	Infrared Senor Module	Detects rotation and registers on Arduino	$8.98	1	$8.98
7	Distance measuring wheel	Used for measuring hill profile distances (not shown)	$39.97	1	$39.97
				Total	$70.67

G Design CAD drawings

Housing



Housing shield



H Prior prototype

The most significant experimental contribution to the program is the estimation of the hill profile; this prototype program simulates the collection of experimental hill slope data and outputs an appropriate hill profile estimation to the user interface. By taking measurements at a constant step size, a straight-line approximation is used to model the hypothetical hill in small pieces. This prototype tests how accurate our hill profile model is to a known function. The chosen function is given by

$$y = \frac{1}{x+1}$$

By averaging two angles, we can determine an approximate θ value to use for our straight-line approximation. A dummy set of data points were calculated using the arc length equation

$$s = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, \mathrm{d}x$$

where $\frac{dy}{dx}$ is the derivative of the chosen function. By setting the arc length constant and solving for the upper bound of the integral, the x values of the dummy data was determined. The slope at these points was found and treated as input to the graphical user interface (GUI) shown in figure 27.

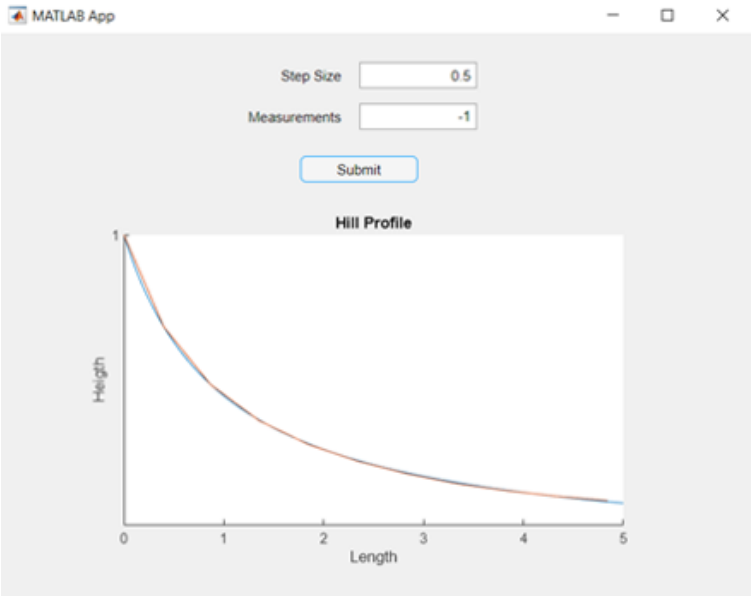


Figure 27: GUI prototype with hill profile estimation and input data fields

A crude implementation of a basic GUI would look something like this. The client would input the constant step size and all the measurements at the corresponding points. The estimated hill profile is generated after every new measurement is added. The blue curve represents our chosen function, and the red line represents the estimated hill profile. The prototype has successfully shown a simple GUI that takes in experimental data from the client and outputs a visual hill profile estimation.

I Velocity data collection procedure

The procedure for collecting velocity data is to make sure that the SD card is inserted into the module, and that the data logger is mounted to the axle with the infrared sensor properly calibrated, as detailed in appendix E. The calibration wheel should be attached to the car with the white tape facing the tachometer device. When the driver is rolling down the hill, they should use the toggle switch to activate the data logger when they have reached a level portion of the hill, and turn the device off just before they apply the brake. Once the data has been collected, the SD card is removed from the logger, and uploaded to the GUI when prompted for "road load data". Each time the data logger is activated, it begins a new datafile. The name of the datafile will follow the format: *DATALOGxx.TXT*, where *xx* is the dataset number (e.g. 0, 1, 2, etc.). These files can be deleted or moved off the SD card when they are no longer needed. The data text files store velocity data in a two-column format: the first column is the data point time stamp (in milliseconds) and the second column is the measured angular velocity of the wheel (in revolutions per second). The raw data can be uploaded to the GUI as is, no post-processing necessary.

J Raw experimental data

Figure 28 and table 8 provide raw angular velocity data recorded from the tachometer device indoors in a lab-setting initial experiment. The calibration was hand-spun up to about 6 rev/s and allowed to coast down over 30 seconds.

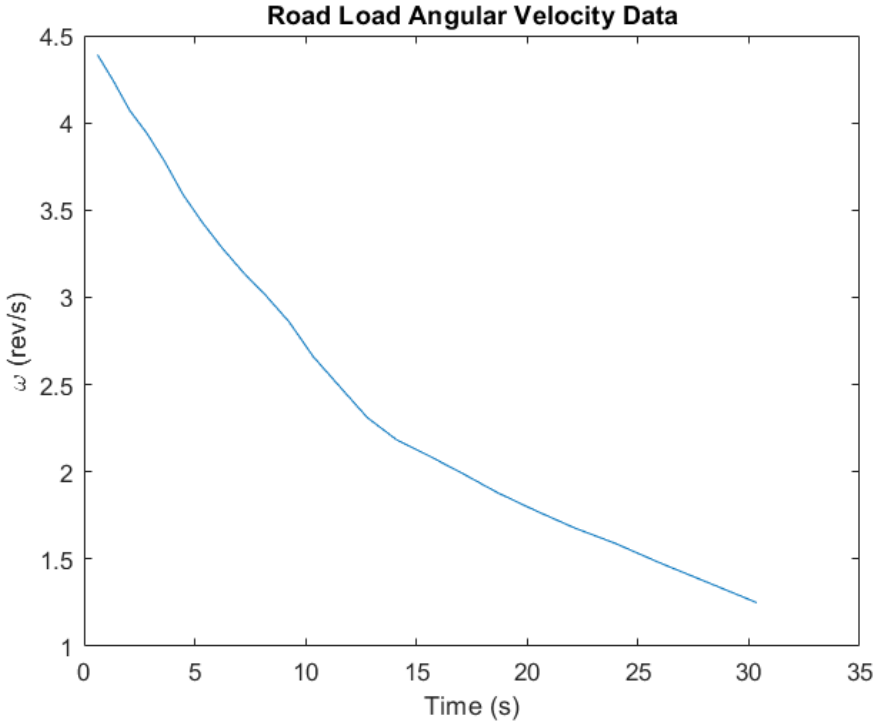


Figure 28: Raw velocity data curve from tachometer testing

Table 8: Raw experimental data from tachometer lab testing

Time (ms)	**Velocity (rev/s)**
0	5.75
611	4.39
1327	4.24
2067	4.07
2837	3.94
3632	3.78
4462	3.59
5335	3.43
6250	3.28
7207	3.14
8206	3.01
9248	2.86
10344	2.66
11525	2.49
12787	2.31
14146	2.18
15584	2.09
17086	1.99
18663	1.88
20329	1.78
22096	1.68
23963	1.59
25943	1.48
28058	1.37
30354	1.25