Rube Goldberg Machine - A Day at the Amusement Park

Trevor Lau
BioE
College of Engineering
Northeastern University
Boston, MA
lau.t@northeastern.edu

Celine Semaan
EECE
College of Engineering
Northeastern University
Boston, MA
semaan.c@northeastern.edu

Alexander Williamson
MechE
College of Engineering
Northeastern University
Boston, MA
williamson.a@northeastern.edu

Jenna Rutberg
BioE
College of Engineering
Northeastern University
Boston, MA
rutberg.j@northeastern.edu

Bala Maheswaran
EECE
College of Engineering
Northeastern University
Boston, MA
mahes@coe.neu.edu

Table of Contents

Rube Goldberg Machine - A Day at the Amusement Park

Abstract

The final project we completed for GE 1501/1502 was a Rube Goldberg Machine titled *A Day at the Amusement Park*. The Rube Goldberg Machine created met all assignment criteria including at least eight steps, five energy transfers, six simple machines, and achieved an end task. In order to recreate an amusement park, we constructed simple machines that resembled common amusement park rides such as rollercoasters, ferris wheels, and merry-go-rounds. Throughout our Rube Goldberg Machine, several scientific concepts were present including conservation of energy and momentum, torque, centripetal acceleration, gravitational force, equilibrium, and friction.

During the construction of each individual machine, we learned a great deal about how simple machines function as well as how to work with the materials and resources at hand. During the final assembly of our Rube Goldberg Machine, we experienced issues with having each machine work flawlessly and consistently with one another based on the materials used as well as trouble with proper function of the sensors and motors. After several trials, we were able to have most of the assembly work at one time. In the future, we believe improvements to the materials used such as incorporating more reliable motors and sensors as well as improvements to the overall setup of our assembly would lead to more consistent results.

Introduction/Background

A Rube Goldberg Machine is a set of several machines that work together to achieve an end task[1]. Throughout the entire Rube Goldberg Machine, energy and momentum are consistently conserved[2] despite the numerous energy transfers that take place. The Rube Goldberg Machine that we designed and constructed follows both these scientific principles. Energy transfers that are present in our machine include transfers from potential energy to kinetic energy, potential energy to kinetic energy to electrical energy to mechanical energy, potential energy to mechanical energy to kinetic energy, and potential energy to electrical energy to kinetic energy. Simple machines present in our Rube Goldberg Machine include the lever, incline plane, wheel and axle, and pulley.

Our Rube Goldberg Machine was based on the recreation of an Amusement Park. The course of the machine begins when the marble is released down the hill and ends at the merry-go-round. Additionally, the course features a loop, drawbridge, ferris wheel, elevator, seesaw, and an autonomous robot car. In order for the end task to be achieved, the released marble goes through the course and triggers different aspects of each machine. The course is complete when the end task of the merry-go-round spinning due to the external force of the autonomous robot car is achieved.

---

[1] What Is a Rube Goldberg Machine?" *Wonderopolis*, National Center for Families Learning, 2020, wonderopolis.org/wonder/what-is-a-rube-goldberg-machine.

[2] "What Is a Rube Goldberg Machine?" *Study.com*, Study.com, 2020, study.com/academy/lesson/what-is-a-rube-goldberg-machine.html.

Design Approach and Methods

Problem Statement

To design and construct a Rube Goldberg machine using at least six simple machines, five energy transfers, and a minimum of eight steps to successfully complete an end task.

Design Approach

Before beginning any design work, we completed general research focused on what a Rube Goldberg Machine is and how one would typically work. After reviewing the requirements of the problem statement, we brainstormed a list of different simple machines that could be implemented, how they could be activated, and several possible end goals. After the generated list of machines was completed, the end goal of spinning a carousel was chosen. The overall theme of "amusement parks" and a number of machines from the brainstormed list were also chosen to be implemented. Research was completed on the relevant scientific principles for each chosen machine. Graphical representations were made to provide a visual representation of a prototype for each machine. Once the graphics were reviewed, a new brainstorm session was held in order to determine the order of each machine in the project, to ensure any constraints on size would be met before construction on the individual machines began. Once each machine was assembled, the machines were tested individually for efficiency, and necessary changes were made. Finally, the machines were brought together to assemble the entire machine - however, due to extenuating circumstances, several parts of the machine were unavailable for the final machine assembly, causing several improvisations to be made. The machine, once constructed, was then tested for efficiency, and minor changes were made which then allowed the machine to complete the end task.

Engineering Design Process & Scientific Concepts

Catapult

Traditionally, a catapult works by utilizing either tension or a torsion bundle to cause an angular acceleration on a lever arm with a payload at the end. The lever arm then hits a stop arm, causing the payload to be launched. Due to the scale of this project, it was determined that a simple class 1 lever would be more practical, both for ease of construction and accuracy of the projectile. This works by dropping an object on one side of the lever, causing a net torque, applying an angular acceleration to the lever until it collides with the ground, at which point the payload would be launched from the opposite end of the lever. After collecting materials to begin construction, it was determined that several changes should be made to the first iteration, namely that the length of the arm should be shortened, and the diameter of the pivot should be increased, to allow the payload to be launched with more horizontal velocity than vertical. This decision was made because the original goal was to have the payload hit a marble down a ramp, and it was determined that this would be more feasible if the payload had more horizontal motion than vertical, as this increases the probability that the payload hits the marble in the intended direction. It was determined that if a .1 kilogram object was released from a height of 1 meter, the payload would travel approximately 4.65m (Appendix B) and this value was relatively close to the estimated experimental value of an object of similar mass from a similar height. Unfortunately, this component of the machine was unable to be implemented, as access to the machine was restricted due to covid protocols. There was also no feasible way to mount the catapult with the materials on hand in a way such that the payload could consistently hit the marble down the ramp.

Loop

Theoretically, a loop is quite a simple machine. The primary issue is getting the desired object to go through the loop without falling off. Before deciding on any sort of measurements, we researched the science behind a loop. After performing several calculations, located in the appendix, the conclusion was quite simple: in order for a rotating object such as a marble to make it through the loop without having any initial velocity, it would have to travel down a hill with a height of 2.7 times the radius of the loop prior to entering the loop. This would provide enough velocity for the marble to not fall off the track. Keeping that in mind, we also had to take into account the size constraints of the machine. It had to be easy to carry around. Therefore, in order to achieve that, we picked a radius of 6 centimeters. This meant that the height had to be of about 16.2 centimeters minimum. To ensure the loop's success, we had to make slight modifications to the theoretical design we had come up with. The minimum height of the hill prior to the loop was increased to allow the marble to make it successfully through the loop while taking into account possible miscalculations and room for error. The hill was placed at a steep angle to help with the velocity and to make sure friction and air resistance were accounted for that could impact the expected velocity of the marble. To construct the design[3], we purchased cardboard and cut the necessary components of the machine using a blade-knife. We then adhered them together using hot glue. For the loop, we achieved a circular shape using cardboard by removing one of the two layers of the cardboard. This allowed it to be extremely flexible and be molded into a circle.

Figure 1A

Drawbridge

The drawbridge is a contraption that serves to both progress the motion of the initial object and simultaneously trigger a chain reaction of events that would proceed independently and work towards achieving the desired end result. As a simple machine, it qualifies as both an inclined plane and a pulley. We selected the drawbridge under the notion that it would provide the final design with additional depth, seeing as it allowed us to reuse portions of the track and run a series of secondary steps parallel to the main course of action.

When compared with the rest of the machine, the overall drawbridge contraption experienced the fewest issues during its development and implementation. The few obstacles that were encountered mostly pertained to the recreation of the setup of the drawbridge's trigger mechanism, which consisted of a pair of carefully balanced marbles that would be sent into motion by a chain reaction of collisions. Due to the nature

[3] Gear, EHC. "How to Make a Hot Wheels Loop Marble Run From Cardboard." *YouTube*, YouTube, 26 June 2018, www.youtube.com/watch?v=Ddt36PoB0sc.

of which these marbles were placed, they often expressed a tendency to roll independently and prematurely initiate the drawbridge's opening sequence. This oversight was mitigated with additional fastening materials and protective rails. No major obstacles were encountered with the motor used to raise and lower the drawbridge, and the amount by which the pulley unwound was determined experimentally.

Very few calculations were necessary in the planning and construction of this device; it should be noted, however, that the pulley used to raise and lower the bridge has a 1:1 input-to-output ratio of forces. The specific dimensions of the drawbridge were considered relatively negligible during the initial stages of planning, and were scaled relative to its adjacent components. The drawbridge was constructed primarily of cardboard, duct tape, rubber bands, and string. Metal hinges were purchased and used in the design to make the rotating action of the bridge as smooth as possible. In terms of electronics, a distance sensor and Arduino were used, both being controlled and powered externally by a laptop. Conveniently, the program used by the drawbridge was simply a modification of the code run by the robot car.

Figure 2A

Figure 2B

<u>Ferris Wheel</u>

A ferris wheel is a type of simple machine known as a wheel & axle. The goals of our design for the ferris wheel were to create a machine that was able to rotate consistently at a slow speed. In our specific design, we used a programmed motor to allow for rotation at a constant speed. Prior to construction, we completed research to better understand how a ferris wheel is built, how it rotates, and the scientific concepts behind the rotation. We learned about the general structure[4] of a ferris wheel as well as that most ferris wheels are rotated by the power from a motor[5]. For the ferris wheel, we created two prototypes and one final design. The first prototype was built out of popsicle sticks and plastic dividers. It was large, and extremely heavy. From this prototype, we learned how to assemble a frame for a ferris wheel, as well as ways to improve the ferris wheel. The second prototype was built on a smaller scale compared to the original prototype. The

---

[4] "The Design Of The Mini Size Ferris Wheel Ride." *Premium Amusement Park & Funfair Ground Rides*, 22 Mar. 2017, amusementrides.org/design-mini-size-ferris-wheel-ride/.

[5] "Observation Wheel Technology." *Observation Wheel Directory*, 7 July 2013, www.observationwheeldirectory.com/ferriswheelarticles/observation-wheel-technology/.

motor, however, was still unable to rotate the axle and in turn rotate the ferris wheel. This meant that the ferris wheel was still too heavy. We continued to research to learn how to better design a ferris wheel, and found that cardboard[6] was a much lighter material that could be used. Taking what we learned from the prototypes and our research, we were able to create a final design. The machine we built was made completely out of cardboard and paper as shown in Figure 3A. The materials used resulted in a much lighter machine.

To allow for the motor to rotate the ferris wheel, the axle of the ferris wheel was connected to the rotating white piece of the motor with a rubber band. The rubber band was pulled tightly, and the motor was secured with velcro onto the base of the ferris wheel. Popsicle sticks were used to raise the base to allow for the proper distance for the rubber band to be pulled as shown in Figure 3B. The motor was connected to the circuit and breadboard, based on project 5A: Motor Basics, located in the SparkFun Inventor's Kit. An Arduino program was then uploaded to the circuit and breadboard from a computer to send a signal to the motor to begin rotating. The program used was the example program SIK_Circuit_A-MotorBasics. The example program was adjusted as the int motorSpeed was changed from 0 to -130. The change from 0 to -130 allowed for the motor speed to constantly be -130 instead of 0 unless a different speed was entered into the serial monitor. The negative sign in the motor speed of -130 refers to the direction of the motor. The ferris wheel successfully rotated. The program and circuit and breadboard diagram is located in Appendix 3.

Based on the ferris wheel's rotation, several observations were taken as shown in table 1 located in Appendix 1. The appropriate calculations were then completed based on these observations, and are located in Appendix 2. The calculations completed found that the average angular velocity of the ferris wheel was 1.838 radians/sec, and the average centripetal acceleration of the ferris wheel was -0.578 m/s^2.

Originally, we had planned to incorporate the consistently rotating ferris wheel discussed above into our final design. The rotating ferris wheel powered by the motor was unable to properly catch the falling marble and carry it down to ground level. The motor was not included in the final design of the ferris wheel, and instead the ferris wheel was gravity-operated. When the marble fell from the drawbridge onto the ferris wheel, the added weight from the marble resulted in the ferris wheel rotating due to the gravitational force and centripetal acceleration. The machine used without the motor is pictured in Figure 3C.

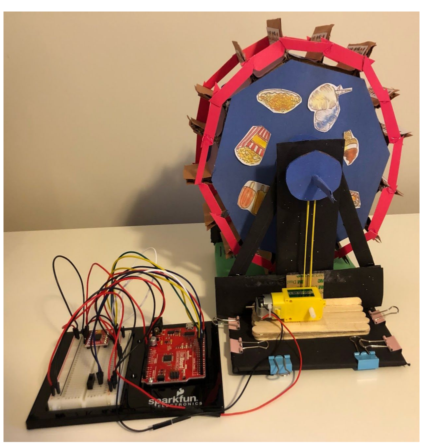


Figure 3A

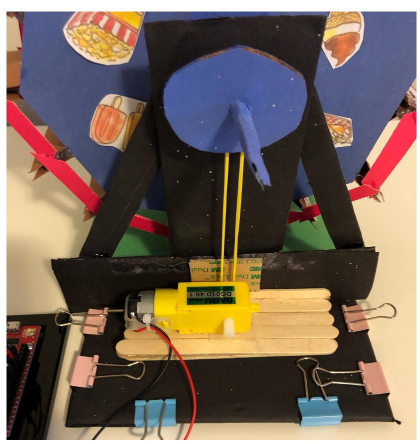


Figure 3B

Figure 3C

[6] "Ferris Wheel Powered By DC Motor." *YouTube*, YouTube, 25 Mar. 2017, www.youtube.com/watch?v=p2kArWM5BaA.

Elevator

The elevator is a deviation from the original contraption, which was intended to be a pulley-operated lift. Due to time constraints, however, development led us towards this iteration of the design. As a simple machine, it qualifies as both a pulley and a lever. This system was chosen because it allowed us to add additional layers to the path taken by the machine, essentially "resetting" the elevation and providing us with more space to work with and opportunities to implement energy transfers.

The primary issue encountered with the initial system was that we often found it difficult to consistently balance and release the marble being held in the lift. This was largely the result of only having a single point of connection with the pulley, which would not have been a problem if we had received the 3D-printed parts on time. We tested a variety of designs to replace the original, but eventually settled on the elevator due to both its consistency and ease of replication. The size of the cardboard box, angle of the applied force, and the height of suspension were also determined experimentally. Another modification that was made was in regards to the contraption's mechanism of activation. The motor was originally meant to be triggered by the marble pressing a button, but testing showed that the marble did not produce enough force to complete that task. As a result, the elevator was changed to utilize a distance sensor, triggered by the motion of the marble leaving the Ferris wheel.

There were relatively few calculations performed in the development of the elevator; it should be noted, however, that the pulley used to raise and lower the bridge has a 1:1 input-to-output ratio of forces. The majority of the optimal dimensions and timing in regards to the design were selected via testing and prototyping. The elevator was constructed primarily of cardboard, duct tape, rubber bands, and string. In terms of electronics, an Arduino was used in addition to the previously mentioned distance sensor, with both being controlled and powered externally by a laptop. Fortunately, the program used by the elevator was simply a modification of the code run by the robot car.

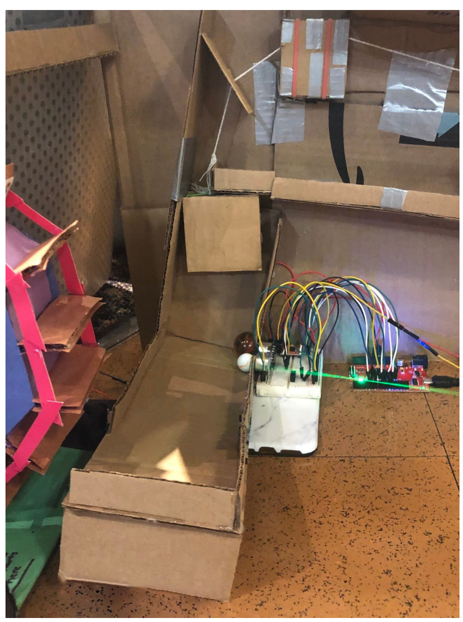


Figure 4A

Seesaw

The seesaw is an improvised substitute for the wedge and box machine which was unable to be used due to restricted access to the needed materials. Originally, marbles were supposed to be added to a cone which would direct pressure onto a button, opening the lid of a box, to activate a distance sensor. The seesaw uses the same principle of adding weight to exceed the force required to lower one side of the seesaw, activating a distance sensor. The improvised contraption also works similarly to the catapult, in that it consists of a lever arm and pivot, but the force required to tilt the seesaw is much less than the force required to launch an object.

Figure 5A

Robot Car

The robot car was built using the instructions guide provided by SparkFun. We built "Circuit 5C: Autonomous Robot". All the components of the robot car came from the SparkFun Inventor's Kit except for the batteries that power the car which we had to purchase ourselves. In order for the machine to have a circular trajectory, we reduced the motor speed of the right wheel leaving the left wheel at normal speed. This allows the robot car to slightly rotate towards the right during its movement. We had to experiment with different motor speed ratios to accommodate for the radius of the merry-go-round. We decided that the robot car would be triggered by the movement of the seesaw contraption that precedes it. Therefore, in order for the car to fully rotate without hitting the other machines located near it, we had to program the car to go forwards for two seconds and then begin its circular motion. We also had to experiment with different ways of attaching a piece of string to the robot car without it getting caught up in the wheel during its movement.

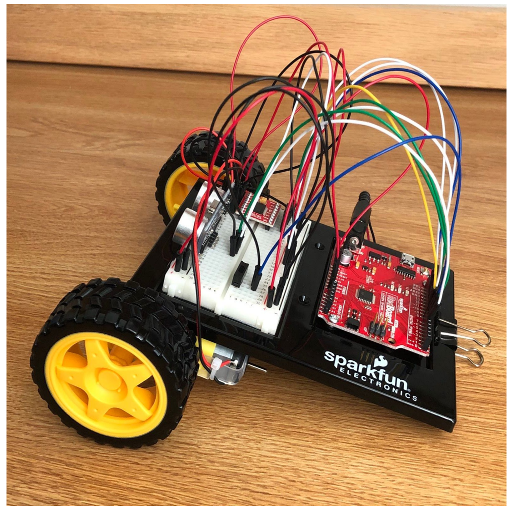


Figure 6A

Merry-Go-Round

A merry-go-round is a type of simple machine known as a wheel & axle. The goals of our design for the merry-go-round was to ensure that it can spin with the help of an external force. In our specific design, the external force was the collision with the attached piece of the autonomous robot car. Prior to construction, we completed research to better understand how a merry-go-round rotates as well as the scientific concepts behind the rotation. We learned that each merry-go-round has a center pole[7] or axle that the rest of the merry-go-round rotates about. As with the ferris wheel, the same theory of physics regarding centripetal force[8] and acceleration apply to the merry-go-round.

The merry-go-round was built using two cardboard circular disks for the top and bottom of the machine. The center of the machine was made out of a cylindrical roll. A center hole was drilled into each disk and the center cylinder. The axle, a wooden cylinder, was then inserted through each center hole with space left on the top and bottom. At this point, the merry-go-round structure was built but the merry-go-round needed to be secured at the top and bottom to allow for rotation and stability. A golf ball was drilled with a center hole, and this hole was aligned with the center hole located on the top disk. The axle was then inserted into the center hole of the golf ball. The top of the axle was now stable. The base of the merry-go-round was an important aspect of the design because it provided stability to the rotation of the merry-go-round. The base of the prototype was not secure, and this resulted in a merry-go-round that was unsteady during rotation. The prototype had a base made out of two stacked cylindrical caps. The caps were not properly secured to the axle and ground. The base of the final design was improved, and allowed for a steadier rotation. The final design base consisted of one cylindrical cap that had a smaller center hole and was adhered to the bottom of the axle. The cylindrical cap was adhered to the ground. The base resulted in an axle of rotation that did not move, and allowed the merry-go-round to successfully rotate steadily due to an external force.

---

[7] "Carousel Construction - How Does a Carousel Work?" *How Carousels Are Made? - How Does a Carousel Work?*, www.historyofcarousels.com/carousel-facts/how-carousels-are-made/.

[8] "Amusement Park Physics: Home Science Tools Learning Center." *Home Science Tools*, 2 Jan. 2020, learning-center.homesciencetools.com/article/amusement-park-physics/.

Figure 7A

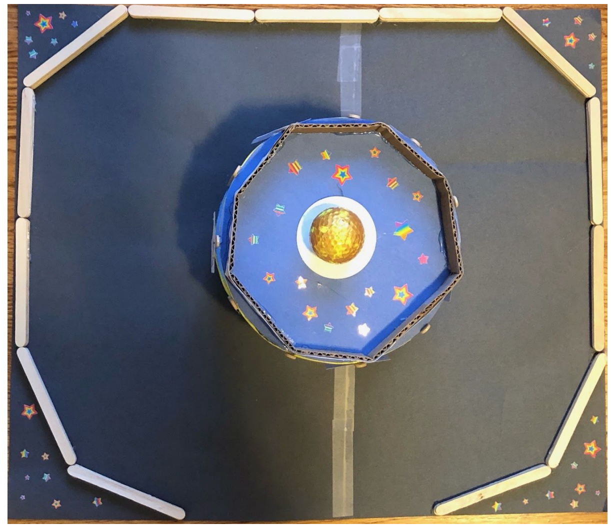


Figure 7B

Graphical Models in 2D & 3D

Catapult

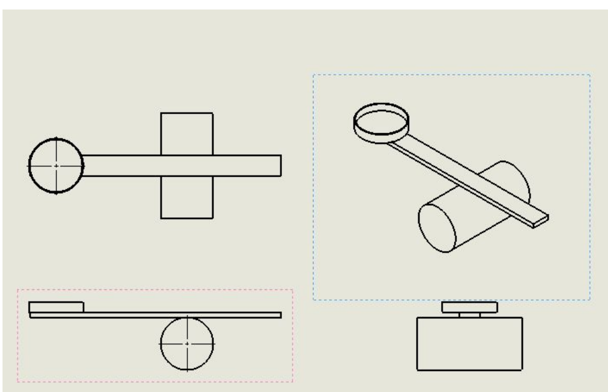


Figure 8A

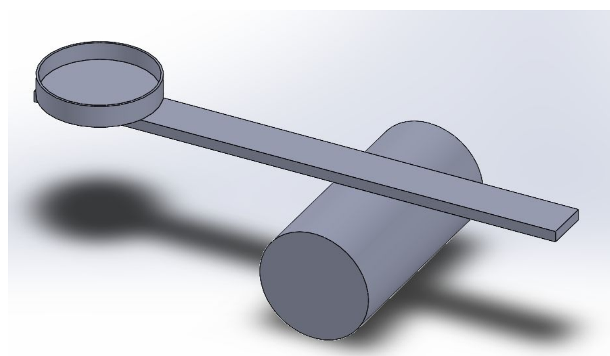


Figure 8B

Loop

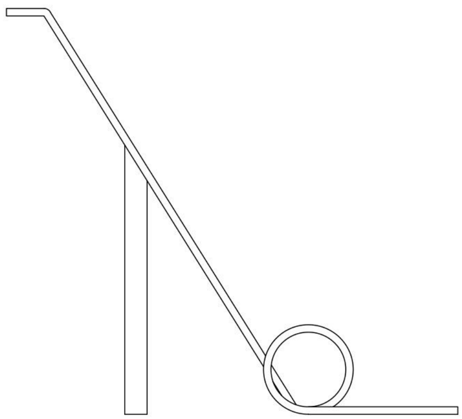


Figure 9A

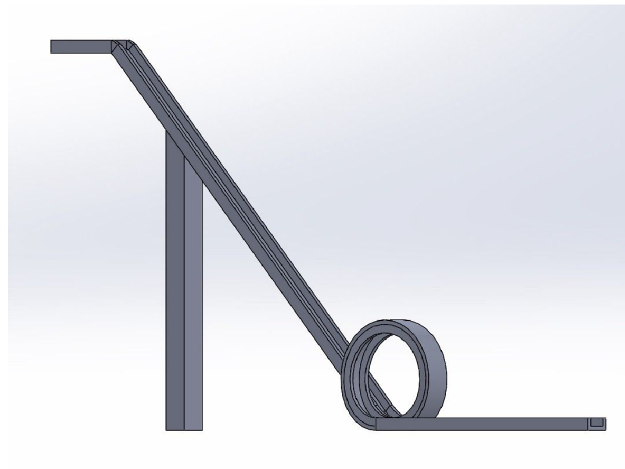


Figure 9B

Drawbridge

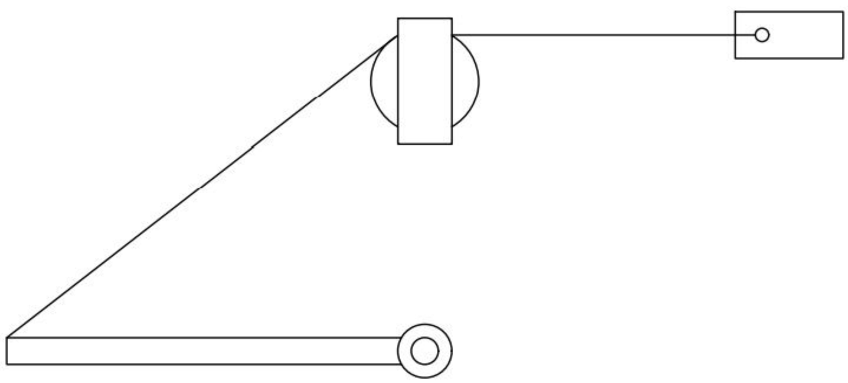


Figure 10A

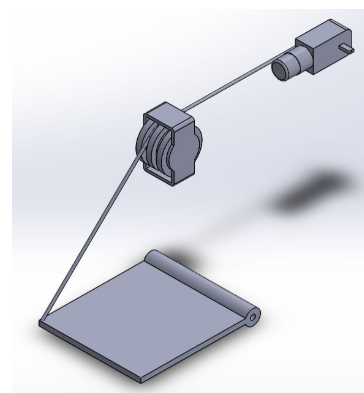


Figure 10B

Ferris Wheel

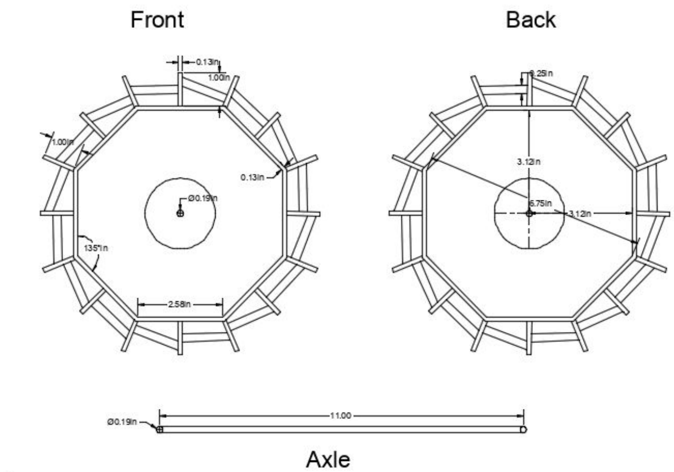


Figure 11A

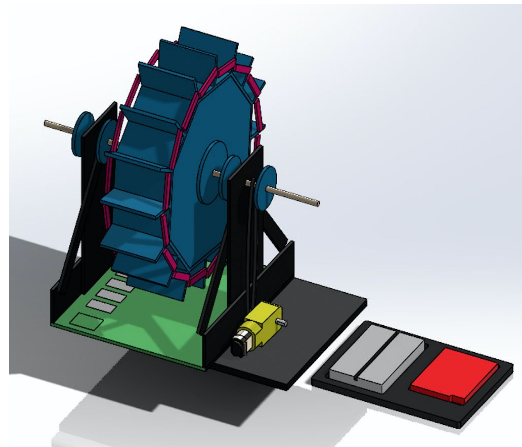


Figure 11B

Elevator

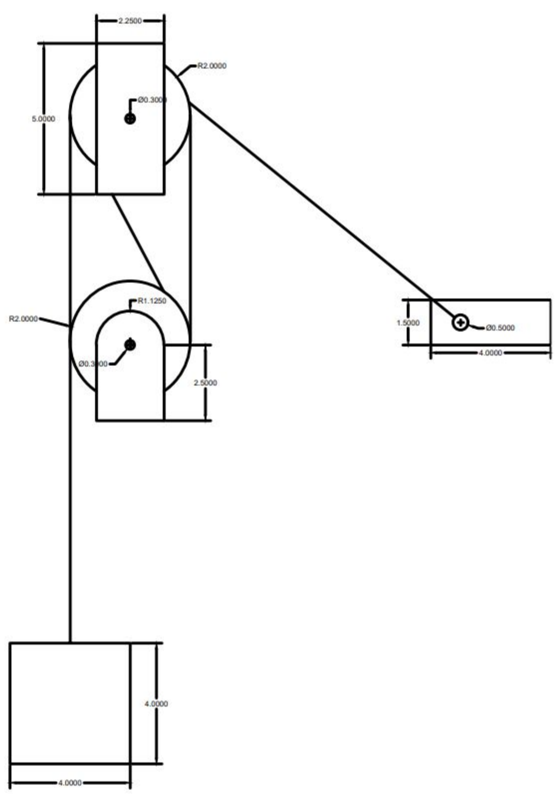


Figure 12A

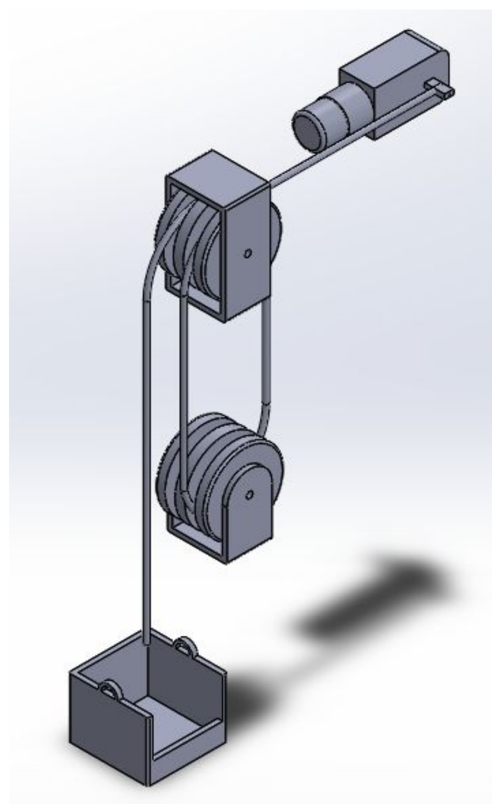


Figure 12B

Seesaw

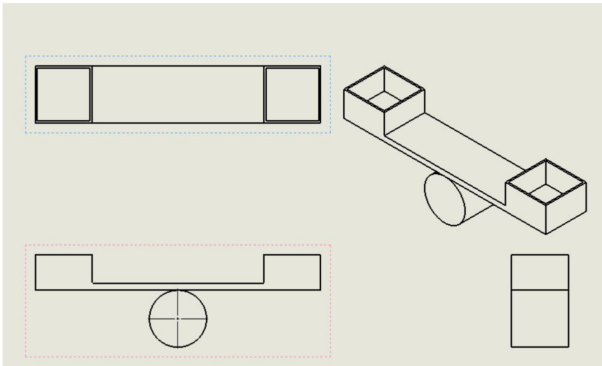


Figure 13A

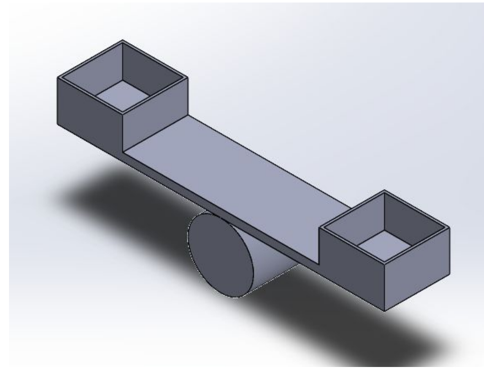


Figure 13B

Robot Car

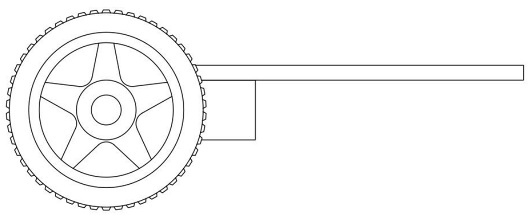


Figure 14A

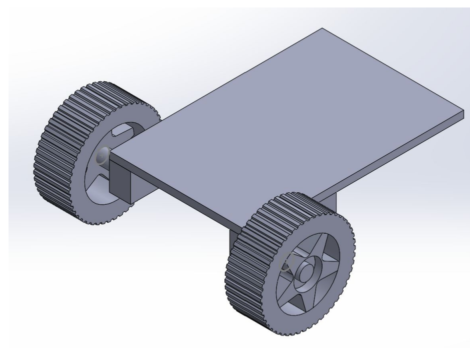


Figure 14B

Merry-Go-Round

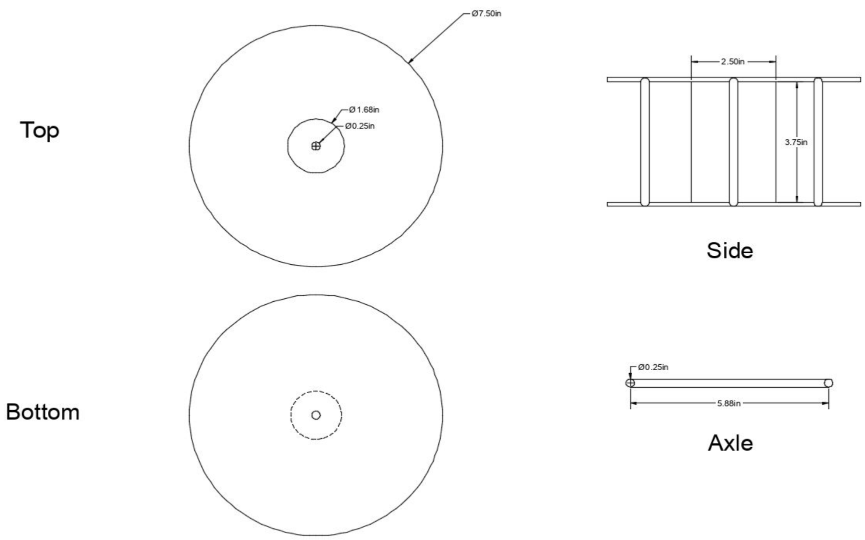


Figure 15A

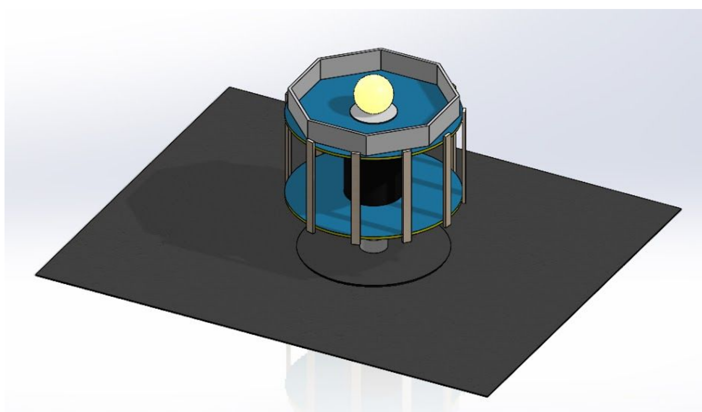


Figure 15B

Design Details

How Does the Rube Goldberg Machine Work?

A marble sits on top of a hill. It is pushed and begins rolling down the hill. It completes the loop and reaches the drawbridge followed by a slight incline. The marble goes up the incline, hits and knocks over a container with a second marble onto the opposite side. The second marble hits a bigger marble that starts rolling and triggers a motion sensor. The previous marble bounces off the latter and drops into a track that leads to the seesaw. Its weight will be added to the contraption to achieve the end goal which will be revealed later on. Back to the first marble. The motion sensor triggers a program that starts the motor for the drawbridge. The drawbridge opens, and the original marble falls through. It then reaches the ferris wheel and rides it until it slides off at ground level. At ground level, the marble rolls towards the elevator, down a track and triggers the distance sensor which starts the motor of the elevator. The elevator begins moving and gets raised until it encounters a section of the track on which a marble is placed. The track tips, and the marble rolls out onto the other section of track which leads to the seesaw. The added weight of the second marble causes the seesaw to fall down and tilt. The distance sensor is triggered by the movement of the seesaw, and a signal is sent to the autonomous robot car which programs it to circle the merry-go-round. The robot car, with a string attached to it on one end and the merry-go-round on the other, begins its circular trajectory around the other machine, causing it to spin. The end task is achieved.

Measurements & How Each Simple Machine Works

Catapult

Measurements:
The arm of the catapult is 30.48cm, with a height of .635cm, and width of 2.54cm. The pivot, with a diameter of 6.35cm, is located 19.05cm from the end of the arm with the payload. The payload cup also has a diameter of 6.35cm, with a height of 1.27cm, with the edge of the cup tangent to the edge of the lever arm (center of cup 3.175cm from edge of lever arm).

How it works:
The catapult sits on a flat surface with the payload resting in the payload cup. An object is dropped onto the other end of the lever arm, causing an angular acceleration until the edge of the arm hits the horizontal surface, at which point the payload is launched from the cup at the linear velocity of the end of the lever arm just prior to the collision of the other end of the arm with the floor.

Loop

Measurements:
The loop has a radius of 6cm. The hill prior to the loop has a height of about 50cm. It is at an approximate 45° angle from the horizontal. The ramp itself is about 70cm. There are sides built on the track that have a height of 1cm.

How it works:
A marble sits on top of the hill and when pushed from behind, it rolls onto the track, into the loop and comes back out.

Drawbridge

Measurements:
The drawbridge utilizes a track with an approximate width of 10cm on both sides of its slope. The contraption also utilizes a pulley with a radius of 2 centimeters.

How it works:
A marble enters the drawbridge from the loop with enough velocity to make its way up a small slope, at the top of which it tips over a container holding a second marble. As a result, the impact pushes the latter down the opposite side of the slope, and sends the former back down to the door of the drawbridge. The second marble then collides with a third, larger, marble before falling down a chute, where it continues along a secondary track for its weight to be added to the seesaw. Now in motion, the third marble begins to roll towards a distance sensor, triggering a motor connected to a pulley that finally lowers the drawbridge. This allows the first marble to proceed onto the rest of the machine.

Ferris Wheel

Measurements:

Ferris Wheel: The frame of the ferris wheel is an octagon with a diameter of 17.145 centimeters. The frame has a circular support piece with a diameter of 5.410 centimeters. The frame and attached circle has a center hole with a diameter of 0.483 centimeters. The distance between the two frames is 5.715 centimeters, and is filled with an internal frame created by rectangular pieces that have the measurements of 5.715 centimeters by 6.502 centimeters. The sixteen vertical divides that are situated above the internal frame have measurements of 2.54 centimeters by 5.715 centimeters. The sixteen railings that are connected to the vertical divides have measurements of 3.429 centimeters by 0.635 centimeters. The axle of the ferris wheel is a cylinder of length of 27.94 centimeters and a diameter of 0.483 centimeters. Cardboard was used to create each piece of the ferris wheel besides the railing. The thickness of the cardboard is 0.330 centimeters.

Base: The base of the ferris wheel has a bottom base that has measurements of 15.570 centimeters by 23.342 centimeters. The top base has measurements of 17.145 centimeters by 15.875 centimeters. The sides of the top base have measurements of 17.145 centimeters by 3.175 centimeters. An axle holder is attached to each side of the top base with measurements of 15.240 centimeters by 6.350 centimeters, and a center hole of diameter 0.483 centimeters. The rectangular support pieces attached to the axle holders have measurements of 12.065 centimeters by 1.27 centimeters. Cardboard was used to create each piece of the base besides the bottom base. The thickness of the cardboard is 0.330 centimeters.

How it works:

The ferris wheel is gravity-operated. Once the marble reaches the ferris wheel after the drawbridge opens, the added weight from the marble results in the rotation of the ferris wheel. The rotation is caused by the gravitational force, centripetal force, and centripetal acceleration acting on the marble and ferris wheel.

Elevator

Measurements:

The track that the marble rests on and the ramp that leads to the distance sensor have widths of 4cm and 10cm respectively. The cardboard box being elevated has dimensions of approximately 6cm x 6cm x 3cm. The contraption also utilizes a pulley with a radius of 2 centimeters.

How it works:

After leaving the Ferris wheel, the marble rolls down a ramp and triggers a distance sensor. The resulting signal causes a motor to wind up a pulley, raising a small cardboard “elevator”. The elevator is initially suspended at rest below a level platform upon which another marble rests. The rising action triggered by the motor results in the elevator displacing the platform, causing the latter to tilt, sending the second marble rolling into the raised basket of the seesaw.

Seesaw

Measurements:

The arm of the seesaw is 20cm long, with a height of .5cm, and width of 4cm. The pivot has a diameter of 4cm, and is located 10cm from either end of the lever arm. The boxes are 4cm by 4cm with a height of 2cm.

How it works:

The seesaw starts with a counterweight in the box on the right side, and then two marbles are added to the box on the left, causing a net torque to apply an angular acceleration on the seesaw, lowering the left side and activating a distance sensor in doing so.

Robot Car

Measurements:
The robot car is about 18 centimeters long and 16 centimeters wide. The height is about 6.5 centimeters (not including the wires connected to the board).

How it works:
A program is uploaded from a computer using the Arduino software to the circuit board that sits on top of the car. The robot car is powered by batteries that are located under it. It is programmed to ride in a circle around another machine: the merry-go-round after being triggered by the movement of the seesaw contraption that precedes it. The purpose of the robot car is to spin the merry-go-round. This is done with the use of a string attached, on one end to the car and the other to the merry-go-round.

Merry-Go-Round

Measurements:
Merry-Go-Round: The top and bottom of the merry-go-round are circles each with a diameter of 19.05 centimeters, and a center hole with a diameter of 0.635 centimeters. The cylinder in the center of the merry-go-round has measurements of 9.525 centimeters tall and a diameter of 6.35 centimeters. The cylinder has a center hole with a diameter of 0.635 centimeters. The golf ball situated above the top frame has a diameter of 4.267 centimeters. The rectangular piece that follows the circumference of the top disc of the merry-go-round is 2.54 centimeters tall. The axle of the merry-go-round is a cylinder with a length of 14.935 centimeters and a diameter of 0.635 centimeters. The base of the merry-go-round is a cylinder with a bottom diameter of 3.81 centimeters, a top diameter of 3.016 centimeters, and a height of 1.589 centimeters. The popsicle stick dividers are spaced about 5.08 centimeters from each other, and at a 45 degree angle from the center. Cardboard was used to make each piece besides the golf ball. The thickness of the cardboard is 0.330 centimeters.
Frame: The merry-go-round structure is located in the center of a frame. The frame has measurements of 54.61 centimeters by 44.45 centimeters.

How it works:
The axle of the merry-go-round is secured on the bottom with a base and on the top with a golf ball. The merry-go-round has the potential to spin. The autonomous robot car is programmed to circle the merry-go-round. A string is connected between the autonomous robot car and the merry-go-round. The constant pull from the autonomous robot car results in the rotation of the merry-go-round.

Results/Outcome

Did it work?

Each of the contraptions, when independently tested and prototyped, generally worked as they were intended to. When integrated together, the resulting Rube Goldberg machine also performed to a satisfactory degree. However, the combination of devices often yielded more inconsistent results than initially observed in the individual machines. Overall, though, the end product can be considered a success, with us being able to record numerous instances of the majority of the components working in conjunction with each other. In addition, we were also able to collect a handful of trials in which the machine operated as smoothly as envisioned during the planning stages of the project.

Overall Scientific Findings

The most prevalent concepts that were demonstrated throughout the Rube Goldberg were the conservation of energy and the conservation of momentum. The latter can be identified in the interactions between objects, such as marbles colliding with one another to generate motion from stillness. The former is the more common form of energy transfer, occurring any time the marbles experience a change in elevation. This conservation of kinetic and gravitational potential energy can be seen in the loop, elevator, Ferris wheel, and all instances of inclined planes in the design. Other scientific concepts observed throughout the development of our machine included but were not limited to: torque, shown by the force exerted at the end of the lever arm of the seesaw, unstable equilibrium, exhibited by the motion of the Ferris wheel as a result of the weight of the marble, and centripetal acceleration, as seen by the rotation of the merry-go-round and Ferris wheel.

Energy Changes

There were four main varieties of energy transfers that could be observed in the Rube Goldberg machine. The first and simplest conversion was the transformation between potential and kinetic energy. This most often consisted of a simple change in elevation or rotation, as seen in the marble making its way through the loop, the rotation of the Ferris wheel due to the weight of the marble and subsequent change in the marble's height, the tilting of the seesaw caused by weight imbalance, and the turning motion of the merry-go-round resulting from the uncoiling action caused by the robot car. The second form of energy transfer demonstrated is kinetic-to-electrical-to-mechanical. An instance of this is a marble moving from rest due to a collision, triggering a distance sensor that sends a signal to start the motor of the drawbridge. Relatedly, the action of the motor releasing the drawbridge at rest can be classified as an instance of potential-to-mechanical-to-kinetic energy transfer. The elevator rising with the help of a motor also falls under this category. The last major form of energy transfer displayed is potential-to-electrical-to-kinetic energy, as exhibited by the robot car, initially at rest, beginning to move due to a signal generated by an activated distance sensor.

Timing and Success

While the planning portion of our machine's development took place over a period of several weeks, the physical construction occurred over a span of twenty-three hours in three days. The overall success rate of the machine was relatively low. While exact numbers were not recorded, final testing required about an hour. Despite this, we were able to obtain a number of successful trials.

Analysis

The whole process went a lot better than expected. Each machine worked very well individually. Initially, building the assembly went smoothly. We were able to successfully improvise and integrate new features of the assembly into the final design. Overall, the construction of the design proved to be structurally sound.

For future improvements, we could develop a more consistent release method for the marble at the beginning. We could also make the set-up for the drawbridge simpler and easier to replicate for consistency. Because we did not get the 3D printed parts on time for the trial, we had some trouble with the pulleys. Therefore, using 3D printed pulley pieces and a pulley machine instead of an elevator could improve the mechanism overall. At the moment, the ferris wheel spins too fast for the marble to successfully and consistently get dropped in. So, we opted for a gravity focused design. For the future, we could work on slowing down the motor that powers the ferris-wheel as well as make the walls higher so that the marble does not fall off of the ferris-wheel after rolling down the track that precedes it. We could additionally work on a more reliable mechanism for attaching the robot car to the merry-go-round such as implementing a track for the string which could prevent it from being caught in the wheels of the car. Finally, we encountered some issues when it came to the motors that powered the pulleys. Therefore, it could be extremely beneficial to invest in more reliable motors to achieve more consistent results overall.

Future Work

Despite the outcome of this project being considered a success, there were still a number of aspects of the development that could have been improved on. If we were to continue work on this project in the future, we would likely do significantly more work towards prototyping more effective designs of the simple machines currently in use, in addition to making more detailed iterations of the 2D and 3D models. These changes would likely mostly be focused around streamlining the design as to produce a larger number of successful trials. The mechanical changes previously mentioned in the analysis would contribute greatly towards accomplishing this.

Conclusion

Ultimately, the design approach of focusing first on building machines which worked individually did allow us to create a Rube Goldberg machine which fulfilled our general goals and requirements, but time constraints limited how much work was done on designing smooth transitions between each machine, as well as the timing between concurrent steps. Adaptations were made based on the materials available, resulting in a final iteration with some previously unconsidered design changes, as a result of the application of the engineering design process mid assembly. If more time had been available, the transitions between machines would have been optimized to reduce energy loss within the system, and more iterations of each individual machine would have been created, to further increase efficiency.

REFERENCES

"Amusement Park Physics: Home Science Tools Learning Center." *Home Science Tools*, 2 Jan. 2020, learning-center.homesciencetools.com/article/amusement-park-physics/.

"Carousel Construction - How Does a Carousel Work?" *How Carousels Are Made? - How Does a Carousel Work?*, www.historyofcarousels.com/carousel-facts/how-carousels-are-made/.

"Catapult Physics." *Real World Physics Problems*, www.real-world-physics-problems.com/catapult-physics.html.

"Ferris Wheel Physics." *Observation Wheel Directory*, 6 June 2013, www.observationwheeldirectory.com/ferriswheelarticles/ferris-wheel-physics/.

"Ferris Wheel Powered By DC Motor." *YouTube*, YouTube, 25 Mar. 2017, www.youtube.com/watch?v=p2kArWM5BaA.

Gear, EHC. "How to Make a Hot Wheels Loop Marble Run From Cardboard." *YouTube*, YouTube, 26 June 2018, www.youtube.com/watch?v=Ddt36PoB0sc.

"How Does a Catapult Work?" sciencing.com, https://sciencing.com/a-catapult-work-4586404.html. 16 November 2020.

Lucas, Jim. "6 Simple Machines: Making Work Easier." *LiveScience*, Purch, 7 Feb. 2018, www.livescience.com/49106-simple-machines.html

"Observation Wheel Technology." *Observation Wheel Directory*, 7 July 2013, www.observationwheeldirectory.com/ferriswheelarticles/observation-wheel-technology/.

"The Design Of The Mini Size Ferris Wheel Ride." *Premium Amusement Park & Funfair Ground Rides*, 22 Mar. 2017, amusementrides.org/design-mini-size-ferris-wheel-ride/.

"What Is a Rube Goldberg Machine?" *Study.com*, Study.com, 2020, study.com/academy/lesson/what-is-a-rube-goldberg-machine.html.

"What Is a Rube Goldberg Machine?" *Wonderopolis*, National Center for Families Learning, 2020, wonderopolis.org/wonder/what-is-a-rube-goldberg-machine.

APPENDIX

Appendix A: Data Collection/ Analysis Using Computers

__Table 1: Ferris Wheel Collected Data__

Trial Number	Total Time (sec)	# of Rotations	Rate of Rotation (rotation/sec)	Angular Velocity (radians/sec)
1	600	141	0.235	1.477
2	300	88	0.293	1.841
3	300	85	0.283	1.778
4	210	67	0.319	2.004
5	120	40	0.333	2.092

Appendix B: Calculations

Catapult Calculations

Range of Projectile:
$v^2 sin(2\theta)/g$

(θ= initial angle of projectile with horizontal)

$\sum$Torque= Iα
ω = (*initial*) + 2(θ)
(α initial will be zero)
(θ = change in angle of ruler w/ horizontal)
v=rω

Torque =rfsin(θ)

Assuming the torques are applied constantly throughout the motion, the sum of the torques is the sum of the integrals of the individual torques applied by the marble, center of mass of the ruler (as the pivot is not at the centerpoint) and dropped mass, from the initial angle to the final angle. (so $\int$(xxxxxxx)dΘ)

The moment of inertia is assumed to be the moment of inertia of a rod about an axis ⅝ from one end, with an assumed mass of .012kg, and the moment of inertia of the marble (.0065kg) as a point mass - giving a Moment of Inertia of ~.004135

Given the mass of the object dropped, v can now be calculated.

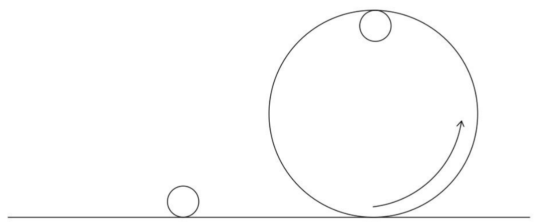
Although the equation given for the range of a projectile is for when the projectile starts at ground level, by having the landing ramp be set at the same height as the catapult cup at launch, no additional work is needed.

While the height the object is dropped at also affects the launch velocity due to conservation of momentum, by keeping the height constant and only changing the mass of the object, we can eliminate this variable, causing the distance calculated to remain a constant factor of the actual distance travelled.

Assuming the mass of the dropped object is .1kg, the projectile travels ~4.65m (seems reasonably close to first tests with an object of estimated mass .1kg dropped from 1m).

<u>**Loop Calculations**</u>

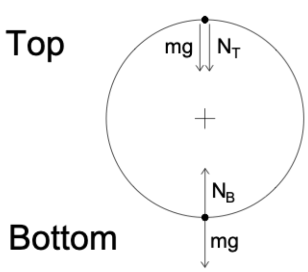
Problem: What is the minimum height a marble needs to start at, if it is to *just* make it around the loop without falling over?



This assumes a friction on the surface, so the ball can ***roll without slipping***.

The first step is to determine the minimum speed the marble will need to just stay on the track when upside down (using centripetal force).
If we can find that minimum speed, then we can use energy conservation to figure out the height at which the marble should start to not fall over when upside down.

Free Body Diagram:



Where mg is the weight,
N_B is the normal force at the bottom of the loop,
N_T is the normal force at the top of the loop.

Forces acting on the marble:

Newton's Second Law of motion: $\sum F_{net = m \cdot a}$

$\sum F_r = mv^2/r$ because $a_c = v^2/r$

Where $\sum F_r$ is the sum of the force in the radial direction (centripetal force).

m is the mass of the marble.
v is the velocity of the marble.
r is the radius of the loop.
a_c is the centripetal acceleration.

Top of the Loop:

$N_T + mg = mv^2/r$
$N_T = mv^2/r - mg$

Bottom of the loop:

$N_B - mg = mv^2/r$
$N_B = mv^2/r + mg$

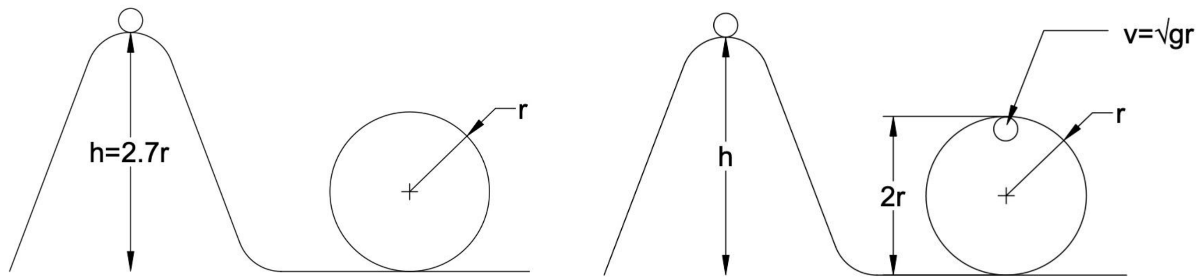
Minimum velocity at the top of the loop for the marble not to fall over:
$N_T = 0$ **;** $v_{min} = ?$

$N_T = mv_{min}{}^2/r - mg = 0$
$mv_{min}{}^2/r = mg$
$v_{min} = \sqrt{gr} \simeq \sqrt{9.81r}$

Minimum height of the hill for the marble to make it through the loop:
We ignore friction.

We know that:
E_g is the potential energy at the top of the hill and $E_g = mgh_{min}$
$E_{g'}$ is the potential energy at the top of the loop and $E_{g'} = mg2r$
$E_{rotational}$ is the rotational energy and $E_{rotational} = 1/2Iw^2$
where $I = 2mr^2/5$ and $w = v/r$

$E_g = E_{g'} + E_k + E_{rotational}$
$mgh_{min} = mg2r + 1/2mv_{min}{}^2 + 1/2Iw^2$
$mgh_{min} = mg2r + 1/2mv_{min}{}^2 + 1/2(2mr^2/5)(v_{min}{}^2/r^2)$
$gh_{min} = g2r + 1/2gr + 1/5gr$
$h_{min} = 2r + ((5 + 2)/10)$
$h_{min} = 2.7r$



<u>Ferris Wheel Calculations</u>

Calculations completed based on collected data in table 1 of Appendix 1.

To find rate of rotation (rotation/sec):

<u>Trial 1:</u>

$141\ rotations/600\ seconds = 0.235\ rotation/sec$

<u>Trial 2:</u>

$88\ rotations/300\ seconds = 0.293\ rotation/sec$

<u>Trial 3:</u>

$85\ rotations/300\ seconds = 0.283\ rotation/sec$

<u>Trial 4:</u>

$67\ rotations/210\ seconds = 0.319\ rotation/sec$

<u>Trial 5:</u>

$40\ rotations/120\ seconds = 0.333\ rotation/sec$

To find average rate of rotation (rotation/sec):

$Average\ rate\ of\ rotation = (Trial\ 1 + Trial\ 2 + Trial\ 3 + Trial\ 4 + Trial\ 5)/5$

$Average\ rate\ of\ rotation = (0.235\ rotation/sec + 0.293\ rotation/sec + 0.283\ rotation/sec +$
$0.319\ rotation/sec + 0.333\ rotation/sec)/5$

$Average\ rate\ of\ rotation =$ **0.2926 rotation/sec**

To find average angular velocity:

$Average\ angular\ velocity = ((Trial\ 1 + Trial\ 2 + Trial\ 3 + Trial\ 4 + Trial\ 5)/5) * 2$

$Average\ angular\ velocity = ((0.235\ rotation/sec + 0.293\ rotation/sec + 0.283\ rotation/sec +$
$0.319\ rotation/sec + 0.333\ rotation/sec)/5) * 2$

$Average\ angular\ velocity =$ **1.838 radians/sec**

To find average centripetal acceleration[9]:

$Average\ centripetal\ acceleration = r * \omega^2 = radius * (angular\ velocity)^2$

$Average\ centripetal\ acceleration = (6.75\ in) * (1\ m/39.37\ in) * (1.838\ rad/sec)^2$

$Average\ centripetal\ acceleration =$ **-0.578** m/s^2

[9] "Ferris Wheel Physics." *Observation Wheel Directory*, 6 June 2013, www.observationwheeldirectory.com/ferriswheelarticles/ferris-wheel-physics/.

**Centripetal acceleration is negative as the force points inward toward the center of the Ferris Wheel*

Appendix C: Programs

Drawbridge Motor Program

drawbridge

```
/*
SparkFun Inventor's Kit
Circuit 5C - Autonomous Robot

View circuit diagram and instructions at: https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40
Download drawings and code at: https://github.com/sparkfun/SIK-Guide-Code
*/

/*
Modifications have been made to the original code to accomadate for a pulley triggered by a distance sensor.
*/


//the left motor will be controlled by the motor B pins on the motor driver
const int PWMB = 10;          //speed control pin on the motor driver for the left motor
const int BIN2 = 9;          //control pin 2 on the motor driver for the left motor
const int BIN1 = 8;          //control pin 1 on the motor driver for the left motor


//distance variables
const int trigPin = 6;
const int echoPin = 5;

int switchPin = 7;             //switch to turn the robot on and off

float distance = 0;            //variable to store the distance measured by the distance sensor

/********************************************************************************/
void setup()
{
  pinMode(trigPin, OUTPUT);       //this pin will send ultrasonic pulses out from the distance sensor
  pinMode(echoPin, INPUT);        //this pin will sense when the pulses reflect back to the distance sensor

  pinMode(switchPin, INPUT_PULLUP);   //set this as a pullup to sense whether the switch is flipped


  //set the motor control pins as outputs
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);


  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(PWMB, OUTPUT);

  Serial.begin(9600);                       //begin serial communication with the computer
  Serial.print("To infinity and beyond!");  //test the serial connection
}

/********************************************************************************/
void loop()
{
  //DETECT THE DISTANCE READ BY THE DISTANCE SENSOR
  distance = getDistance();

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" in");              // print the units

  if(digitalRead(switchPin) == LOW){  //if the on switch is flipped

    if(distance>=3){                //if no object is detected then motor stops
      Serial.print(" ");
      Serial.print("Not Moving...");

      leftMotor(0);

    }else{                         //if an object is detected then motor starts
      Serial.print(" ");
      Serial.print("Moving...");

      leftMotor(255);

      delay(500);

      leftMotor(0);
      while(1);
    }
  }

  else{                          //if the switch is off then stop
```

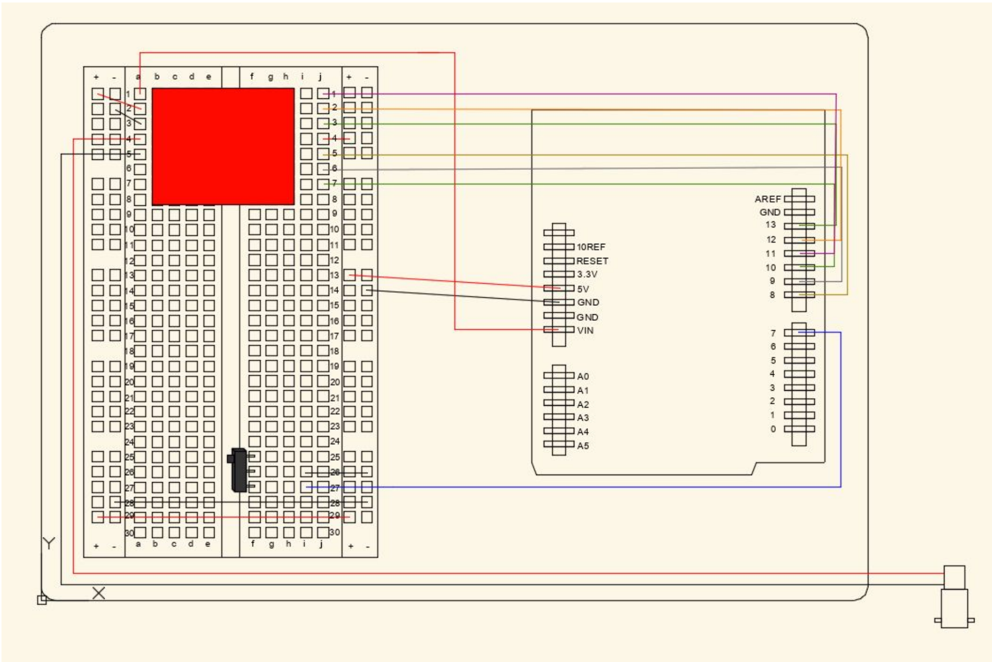
```
      //stop the motors
      leftMotor(0);
  }
}

/********************************************************************************/
//FUNCTION FORDRIVING LEFT MOTOR
void leftMotor(int motorSpeed)
{
  if (motorSpeed > 0)                                 //if the motor should drive forward (positive speed)
  {
    digitalWrite(BIN1, HIGH);                         //set pin 1 to high
    digitalWrite(BIN2, LOW);                          //set pin 2 to low
  }
  else                                                //if the motor should stop
  {
    digitalWrite(BIN1, LOW);                          //set pin 1 to low
    digitalWrite(BIN2, LOW);                          //set pin 2 to low
  }
  analogWrite(PWMB, abs(motorSpeed));                 //now that the motor direction is set, drive it at the entered speed
}

/********************************************************************************/
//RETURNS THE DISTANCE MEASURED BY THE HC-SR04 DISTANCE SENSOR
float getDistance()
{
  float echoTime;                   //variable to store the time it takes for a ping to bounce off an object
  float calculatedDistance;         //variable to store the distance calculated from the echo time

  //send out an ultrasonic pulse that's 10ms long
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime = pulseIn(echoPin, HIGH);      //use the pulsein command to see how long it takes for the
                                          //pulse to bounce back to the sensor

  calculatedDistance = echoTime / 148.0;  //calculate the distance of the object that reflected the pulse (half the bounce time multiplied by the speed of sound)

  return calculatedDistance;              //send back the distance that was calculated
}
```

Ferris Wheel Motor Program

Program originated from https://github.com/sparkfun/SIK-Guide-Code, but a change was made to the int motorSpeed from 0 to -130.

```
//PIN VARIABLES
//the motor will be controlled by the motor A pins on the motor driver
const int AIN1 = 13;           //control pin 1 on the motor driver for the right motor
const int AIN2 = 12;            //control pin 2 on the motor driver for the right motor
const int PWMA = 11;            //speed control pin on the motor driver for the right motor

int switchPin = 7;             //switch to turn the robot on and off

//VARIABLES
int motorSpeed = -130;       //starting speed for the motor

void setup() {
  pinMode(switchPin, INPUT_PULLUP);   //set this as a pullup to sense whether the switch is flipped

  //set the motor control pins as outputs
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

  Serial.begin(9600);                       //begin serial communication with the computer

  Serial.println("Enter motor speed (0-255)... ");    //Prompt to get input in the serial monitor.
}

void loop() {

  if (Serial.available() > 0) {         //if the user has entered something in the serial monitor
    motorSpeed = Serial.parseInt();     //set the motor speed equal to the number in the serial message

    Serial.print("Motor Speed: ");      //print the speed that the motor is set to run at
    Serial.println(motorSpeed);
  }

  if (digitalRead(7) == LOW) {          //if the switch is on...
    spinMotor(motorSpeed);
  } else {                              //if the switch is off...
    spinMotor(0);                   //turn the motor off
  }


}

/********************************************************************************/
void spinMotor(int motorSpeed)                       //function for driving the right motor
{
  if (motorSpeed > 0)                                 //if the motor should drive forward (positive speed)
  {
    digitalWrite(AIN1, HIGH);                         //set pin 1 to high
    digitalWrite(AIN2, LOW);                          //set pin 2 to low
  }
  else if (motorSpeed < 0)                            //if the motor should drive backward (negative speed)
  {
    digitalWrite(AIN1, LOW);                          //set pin 1 to low
    digitalWrite(AIN2, HIGH);                         //set pin 2 to high
  }
  else                                                //if the motor should stop
  {
    digitalWrite(AIN1, LOW);                          //set pin 1 to low
    digitalWrite(AIN2, LOW);                          //set pin 2 to low
  }
  analogWrite(PWMA, abs(motorSpeed));                 //now that the motor direction is set, drive it at the entered speed
}
```

__Ferris Wheel Motor Circuit & Breadboard__



Elevator Motor Program

elevator

```
/*
  SparkFun Inventor's Kit
  Circuit 3B-Distance Sensor
  Control the color of an RGB LED using an ultrasonic distance sensor.
  This sketch was written by SparkFun Electronics, with lots of help from the Arduino community.
  This code is completely free for any use.
  View circuit diagram and instructions at: https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v41
  Download drawings and code at: https://github.com/sparkfun/SIK-Guide-Code
*/

const int trigPin = 6;           //connects to the trigger pin on the distance sensor
const int echoPin = 5;           //connects to the echo pin on the distance sensor

const int AIN1 = 13;
const int AIN2 = 12;
const int PWMA = 11;

float distance = 0;               //stores the distance measured by the distance sensor

void setup()
{
  Serial.begin (9600);        //set up a serial connection with the computer

  pinMode(trigPin, OUTPUT);   //the trigger pin will output pulses of electricity
  pinMode(echoPin, INPUT);    //the echo pin will measure the duration of pulses coming back from the distance sensor
}

void loop() {
  distance = getDistance();   //variable to store the distance measured by the sensor

  Serial.print(distance);     //print the distance that was measured
  Serial.println(" in");      //print units after the distance

  if (distance < 2) {
    bridge(255);
    }

  else {
    bridge(0);
  }
}

//------------------FUNCTIONS-------------------------------

//RETURNS THE DISTANCE MEASURED BY THE HC-SR04 DISTANCE SENSOR
float getDistance()
{
  float echoTime;                   //variable to store the time it takes for a ping to bounce off an object
  float calculatedDistance;         //variable to store the distance calculated from the echo time

  //send out an ultrasonic pulse that's 10ms long
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime = pulseIn(echoPin, HIGH);      //use the pulsein command to see how long it takes for the
                                          //pulse to bounce back to the sensor

  calculatedDistance = echoTime / 148.0;  //calculate the distance of the object that reflected the pulse (half the bounce time multiplied by the speed of sound)

  return calculatedDistance;              //send back the distance that was calculated
}

void bridge(int motorSpeed)                       //function for driving the right motor

{
  if (motorSpeed > 0)                                 //if the motor should drive forward (positive speed)
  {
    digitalWrite(AIN1, HIGH);                         //set pin 1 to high
    digitalWrite(AIN2, LOW);                          //set pin 2 to low
  }
  else                                                //if the motor should stop
  {
    digitalWrite(AIN1, LOW);                          //set pin 1 to low
    digitalWrite(AIN2, LOW);                          //set pin 2 to low
  }
  analogWrite(PWMA, abs(motorSpeed));                 //now that the motor direction is set, drive it at the entered speed
}
```

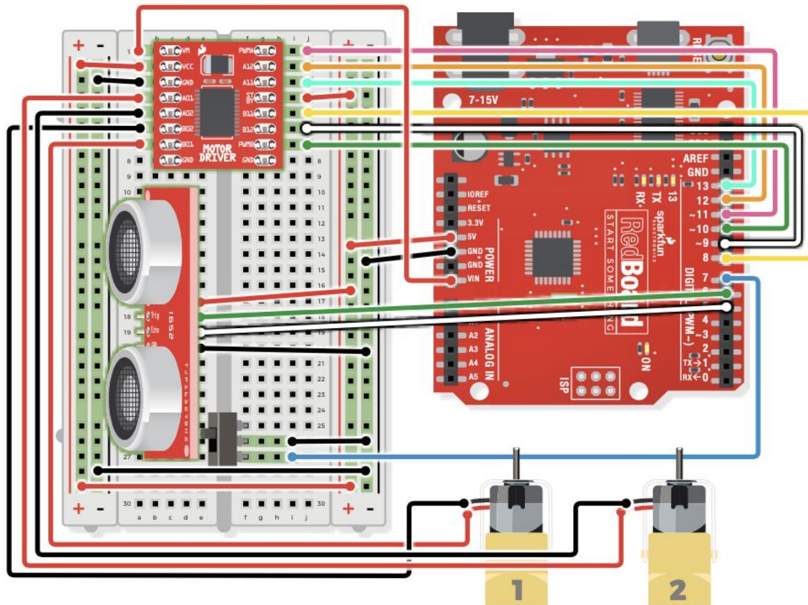
Autonomous Robot Car Program

car

```
/*
SparkFun Inventor's Kit
Circuit 5C - Autonomous Robot

View circuit diagram and instructions at: https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40
Download drawings and code at: https://github.com/sparkfun/SIK-Guide-Code
*/

/*
Modifications have been made to the original code to allow for the robot car to move in a circular trajectory after being
triggered by the distance sensor.
*/


//the right motor will be controlled by the motor A pins on the motor driver
const int AIN1 = 13;           //control pin 1 on the motor driver for the right motor
const int AIN2 = 12;            //control pin 2 on the motor driver for the right motor
const int PWMA = 11;            //speed control pin on the motor driver for the right motor

//the left motor will be controlled by the motor B pins on the motor driver
const int PWMB = 10;           //speed control pin on the motor driver for the left motor
const int BIN2 = 9;           //control pin 2 on the motor driver for the left motor
const int BIN1 = 8;           //control pin 1 on the motor driver for the left motor


//distance variables
const int trigPin = 6;
const int echoPin = 5;

int switchPin = 7;             //switch to turn the robot on and off

float distance = 0;            //variable to store the distance measured by the distance sensor

/********************************************************************************/
void setup()
{
  pinMode(trigPin, OUTPUT);       //this pin will send ultrasonic pulses out from the distance sensor
  pinMode(echoPin, INPUT);        //this pin will sense when the pulses reflect back to the distance sensor

  pinMode(switchPin, INPUT_PULLUP);   //set this as a pullup to sense whether the switch is flipped


  //set the motor control pins as outputs
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(PWMA, OUTPUT);

  pinMode(BIN1, OUTPUT);
  pinMode(BIN2, OUTPUT);
  pinMode(PWMB, OUTPUT);

  Serial.begin(9600);                       //begin serial communication with the computer
  Serial.print("To infinity and beyond!");  //test the serial connection
}

/********************************************************************************/
void loop()
{
  //DETECT THE DISTANCE READ BY THE DISTANCE SENSOR
  distance = getDistance();

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" in");              // print the units

  if(digitalRead(switchPin) == LOW){  //if the on switch is flipped

    if(distance>=4){                //if an object is detected then car stops
      Serial.print(" ");
      Serial.print("Not Moving...");

      rightMotor(0);
      leftMotor(0);

    }else{                         //if no obstacle is detected drive forward
      Serial.print(" ");
      Serial.print("Moving...");

      rightMotor(150);
      leftMotor(150);

      delay(2000);
```

```
      rightMotor(100);
      leftMotor(150);

      while(1);

    }
  }

  else{                         //if the switch is off then stop

      //stop the motors
      rightMotor(0);
      leftMotor(0);
  }
}

/********************************************************************************/
//FUNCTION FORDRIVING RIGHT MOTOR
void rightMotor(int motorSpeed)
{
  if (motorSpeed > 0)                                 //if the motor should drive forward (positive speed)
  {
    digitalWrite(AIN1, HIGH);                         //set pin 1 to high
    digitalWrite(AIN2, LOW);                          //set pin 2 to low
  }
  else                                                //if the motor should stop
  {
    digitalWrite(AIN1, LOW);                          //set pin 1 to low
    digitalWrite(AIN2, LOW);                          //set pin 2 to low
  }
  analogWrite(PWMA, abs(motorSpeed));                 //now that the motor direction is set, drive it at the entered speed
}


/********************************************************************************/
//FUNCTION FORDRIVING LEFT MOTOR
void leftMotor(int motorSpeed)
{
  if (motorSpeed > 0)                                 //if the motor should drive forward (positive speed)
  {
    digitalWrite(BIN1, HIGH);                         //set pin 1 to high
    digitalWrite(BIN2, LOW);                          //set pin 2 to low
  }
  else                                                //if the motor should stop
  {
    digitalWrite(BIN1, LOW);                          //set pin 1 to low
    digitalWrite(BIN2, LOW);                          //set pin 2 to low
  }
  analogWrite(PWMB, abs(motorSpeed));                 //now that the motor direction is set, drive it at the entered speed
}

/********************************************************************************/
//RETURNS THE DISTANCE MEASURED BY THE HC-SR04 DISTANCE SENSOR
float getDistance()
{
  float echoTime;                   //variable to store the time it takes for a ping to bounce off an object
  float calculatedDistance;         //variable to store the distance calculated from the echo time

  //send out an ultrasonic pulse that's 10ms long
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  echoTime = pulseIn(echoPin, HIGH);      //use the pulsein command to see how long it takes for the
                                          //pulse to bounce back to the sensor

  calculatedDistance = echoTime / 148.0;  //calculate the distance of the object that reflected the pulse (half the bounce time multiplied by the speed of sound)

  return calculatedDistance;              //send back the distance that was calculated
}
```

Autonomous Robot Car Circuit Diagram



JUMPER WIRES	5V to 5V; GND to GND (-); VIN to A1; D8 to J5; D9 to J6; D10 to J7; D11 to J1; D12 to J2; D6 to E18; D5 to E19; D13 to J3; D7 to I27; 5V (+) to 5V (+); GND (-) to GND (-); A2 to 5V (+); A3 to GND (-); J4 to 5V (+); I26 to GND (-); E17 to 5V (+); E20 to GND (-)
MOTOR 1 (RIGHT)	A4(RED +); A5(BLACK -)
MOTOR 2 (LEFT)	A7(RED +); A6(BLACK -)
MOTOR DRIVER	C1-C8 to G1-G8 *(VM on C1, PWMA on G1)*
SWITCH	F25 + F26 + F27
DISTANCE SENSOR	A17(VCC) + A18(TRIG) + A19(ECHO) + A20(GND)