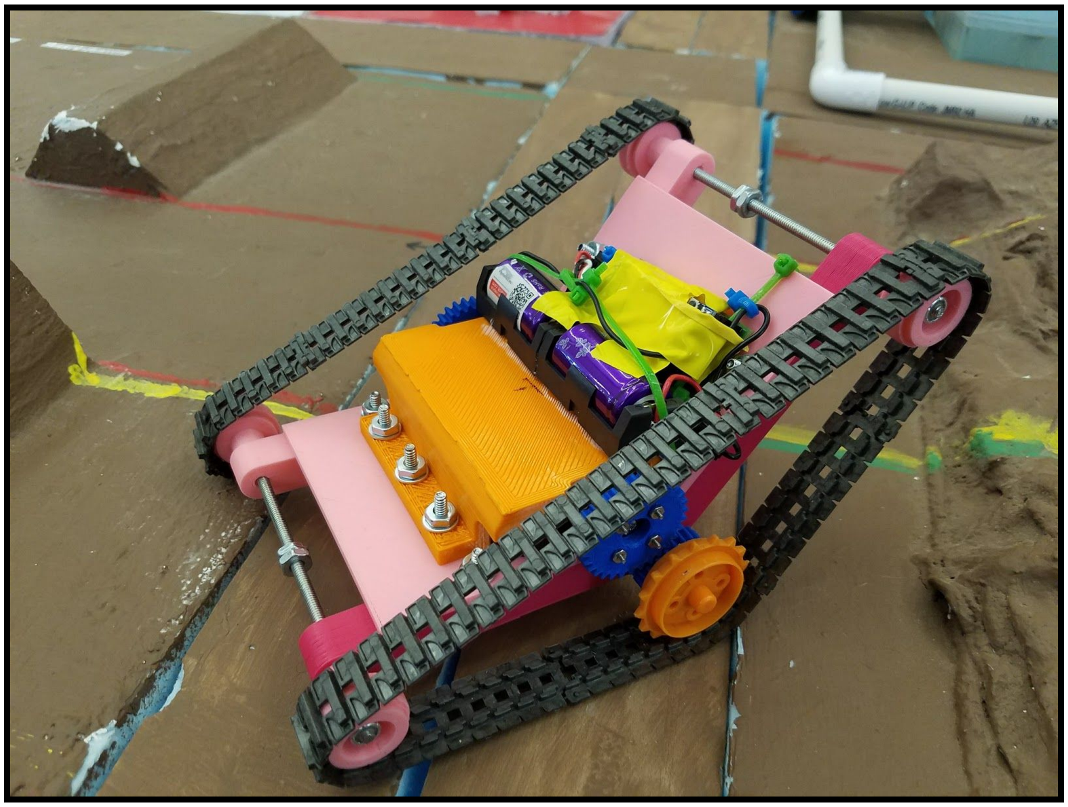
Introducing Humphrey: A Treaded, Flippable Vehicle Capable of Solving the Cat's Conundrum



Kyle Hagerman - Computer Science, Ryan Benge - Chemical Engineering, Ivan Albert - Mechanical Engineering, Michael Price - Electrical Engineering, Berke Gunaydin - Civil Engineering
Montana State University
April 25, 2018

Introducing Humphrey: A Treaded, Flippable Vehicle Capable of Solving the Cat's Conundrum

Kyle Hagerman - Computer Science, Ryan Benge - Chemical Engineering, Ivan Albert - Mechanical Engineering, Michael Price - Electrical Engineering, Berke Gunaydin - Civil Engineering
Montana State University
April 25, 2018

Summary

Humphrey is our solution to Montana State University's general engineering project, the Cat's Conundrum. The Cat's Conundrum itself is a variable obstacle course that must be completed by some form of vehicle. With regards to various size restrictions and project parameters our multidisciplinary engineering team designed Humphrey to complete the obstacle course. Also included in the problem design are two obstacle based challenges, The Swamp and The Embankment. Our team worked diligently and cohesively to solve the problem. Multiple design and test phases were used to narrow down the exact solutions used.

Contents

Introduction

The purpose of this project was to teach us, as discipline specific students, to work in a multidisciplinary setting. Cooperating in this way will prepare each of us for a future work environment. Learning how to design system components for integration as a whole unit is vital in working as a team. The Cat's Conundrum is more than sufficient in testing this skill.

Our team was required to design a remote control vehicle to traverse an obstacle course as well as design a pump system and an embankment which our vehicle would also have to cross. Each discipline was represented fairly and also relied on the outcomes of each team member's contribution. Learning multidisciplinary design is the sole purpose of EGEN 310. This report details our solution and how we arrived at such through various designing and testing stages.

Overview of Multidisciplinary Design

Our team was composed of five members. Represented majors of study include: Mechanical Engineering, Computer Science, Electrical Engineering, Chemical Engineering, and Civil Engineering. The inclusion or exclusion of certain fields in our team's make up determined the team's restrictions on building our remote control car. For example, we have to complete The Swamp because we have a Chemical Engineer on the team.

Introduction of Team: Design Responsibilities

Ivan Albert - Mechanical Engineering
Design Responsibilities: Ivan was in charge of the overall design of the vehicle and the fabrication of the physical structure. He created the vehicle concept for tackling the course obstacles and modeled and fabricated the gears, wheels and frame. He also found many of the purchased parts such as the treads used in the design.
Strengths/Skills: Ivan brought to the team his extensive knowledge of SolidWorks and his familiarity with 3D printers. He also applied his knowledge of physics to find a simple design that's shape and weight distribution allowed it to flip into different configurations to solve a lot of course problems with minimal moving parts.

Michael Price - Electrical Engineering
Design Responsibilities: Design the power delivery, energy storage, circuit hardware layout, and selection of the driving motors. Selection of the electronics and motors was based on specifications set forth in the course rules and by other disciplines on this project. As such, every component was heavily vetted to meet the various needs of all team members.
Strengths/Skills: Several years as an unofficial electronics technician, involving design and repair of bulk liquid monitoring systems. Experience with RF communication basics and program integration from analog sensor/control device networks into a Windows

environment. Very adaptable to changing conditions and highly experienced in last second or temporary modifications to make a design function.

Ryan Benge - Chemical Engineering
Design Responsibilities: Create enough head loss in a two foot section of piping that creates steady state between a high flow pump and a low flow pump.
Strengths/Skills: Enough background with fluid mechanics to create a mathematical model for the losses needed. Able to come up with solutions to put these losses into effect in the design.

Berke Gunaydin - Civil Engineering
Design Responsibilities: The embankment turns 90 degree with the maximum angle and also it fits into 2x2 square ft area. The embankment have to carry the vehicle without breaking.
Strengths/Skills: The embankment prevents the car from sliding. The embankment carries the vehicle from the track to water pump.

Kyle Hagerman - Computer Science
Design Responsibilities: Program the hardware with an onboard
microcontroller and also produce a graphical user interface on a platform of choice to control the vehicle.
Strengths/Skills: Can adapt to new programming languages quickly.
Understands Object Oriented Design and uses control structures to control method calling.
Takes initiative and maintains constant communication with team members.

Background Information and Physical Concepts Underlying the Design

The overall concept for our design was to create a vehicle that could approach a multitude of problems simply by flipping itself over. Humphrey accomplished this with tank treads oriented in the shape of an isosceles triangle. With this tread profile, all other components could be placed within the bounds of the treads so that Humphrey can drive on any tread surface. It was also important that the center of mass be located as close to the driving axle of the vehicle as possible. This allowed Humphrey to have its treads raised way out in front to climb over obstacles, to vary position for maximum stability when going both up and down slopes, and to be a wedge for pushing through the weighted gate. The only moving parts required for this versatility were the driving mechanism.

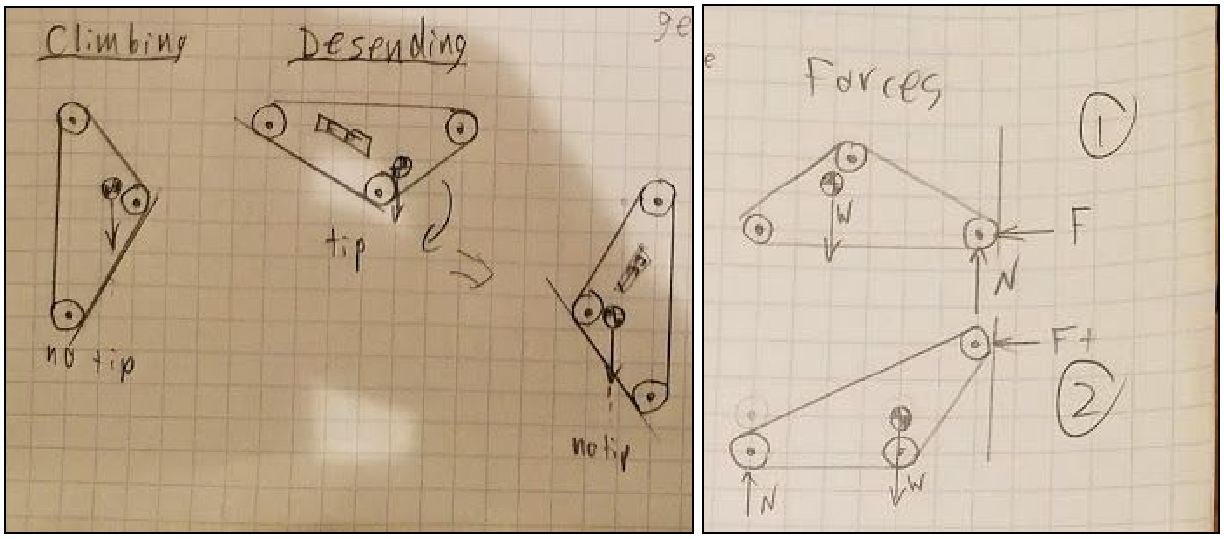


Figure 1-2. Early Concept drawings of the triangle-tank design and the properties of each orientation based on the center of mass.

Design Functions, Objectives, and Constraints

To help aid the design process, our team used tools such as the Objectives Tree (Figure 3) and Functional Diagram (Figure 4). The Objectives Tree helped to lay out what we wanted the vehicle to accomplish. Working towards completing individual tasks compartmentalized the workload and allowed for us to assign individual team member focus as needed. The Functional Diagram also compartmentalized tasks. Viewing this design stage from two perspectives allowed us to build a firm grasp of what work was required to complete our vehicle.

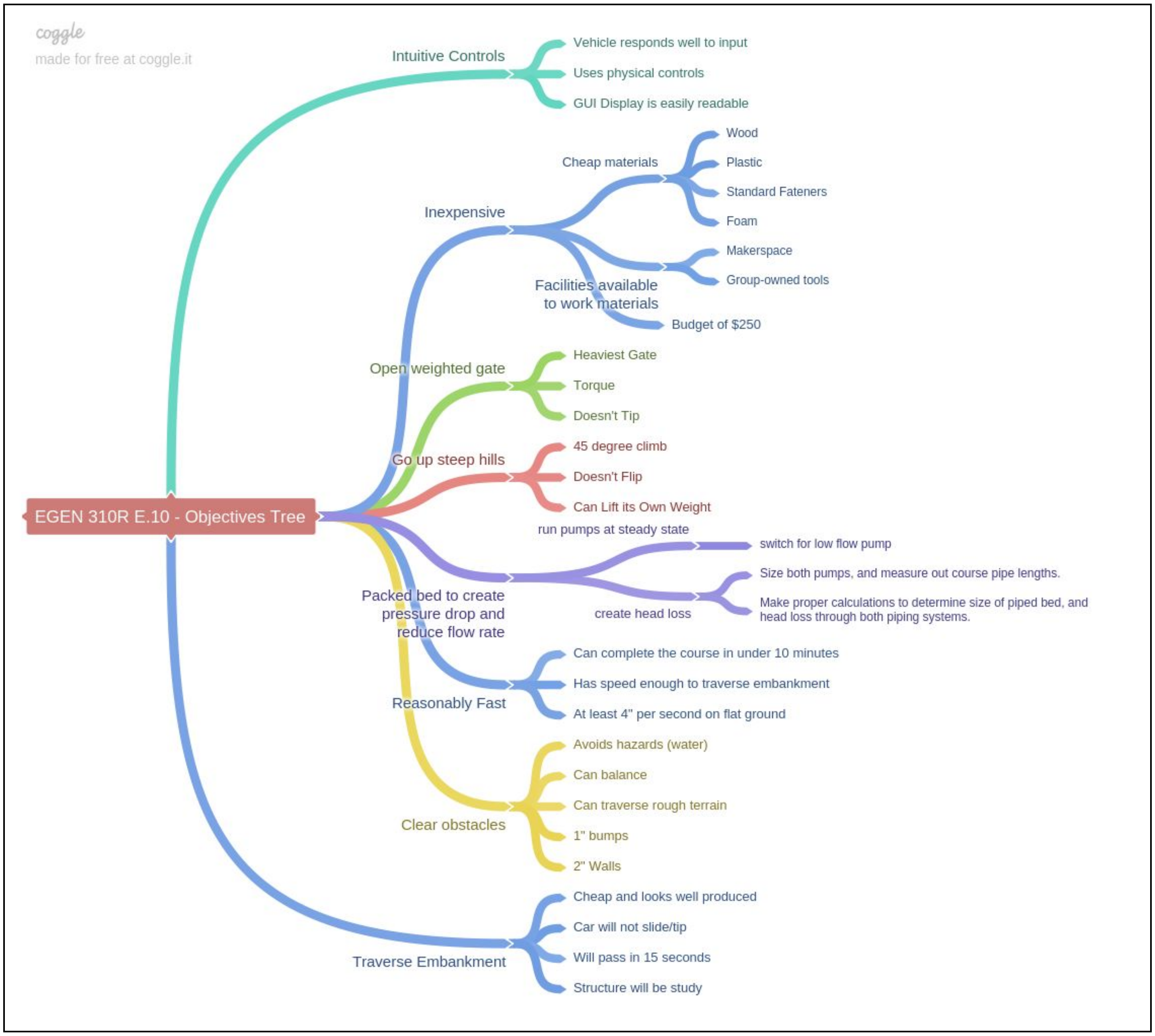

coggle
made for free at coggle.it
EGEN 310R E.10 - Objectives Tree
Intuitive Controls
Vehicle responds well to input
Uses physical controls
GUI Display is easily readable
Inexpensive
Cheap materials
Wood
Plastic
Standard Fateners
Foam
Facilities available to work materials
Makerspace
Group-owned tools
Budget of $250
Open weighted gate
Heaviest Gate
Torque
Doesn't Tip
Go up steep hills
45 degree climb
Doesn't Flip
Can Lift its Own Weight
Packed bed to create pressure drop and reduce flow rate
run pumps at steady state
switch for low flow pump
create head loss
Size both pumps, and measure out course pipe lengths.
Make proper calculations to determine size of piped bed, and head loss through both piping systems.
Reasonably Fast
Can complete the course in under 10 minutes
Has speed enough to traverse embankment
At least 4" per second on flat ground
Clear obstacles
Avoids hazards (water)
Can balance
Can traverse rough terrain
1" bumps
2" Walls
Traverse Embankment
Cheap and looks well produced
Car will not slide/tip
Will pass in 15 seconds
Structure will be study

Figure 3. Our Objectives Tree displays vital objectives. First branches mark a different objective, further branches of the same color break the topic into tasks to focus on. The chart was made with Coggle, a free online flowchart designer.

Our main concerns regarding the vehicle were its ability to climb and to clear obstacles. Our tree will show that by traversing rough terrain we can clear more obstacles. Having good balance and speed help to traverse the embankment. As a team, we decided to complete the heaviest gate on the weighted gate element. Our flippable design was key in our conversation to finish that element. Our vehicle had to be light so we would require higher torque.

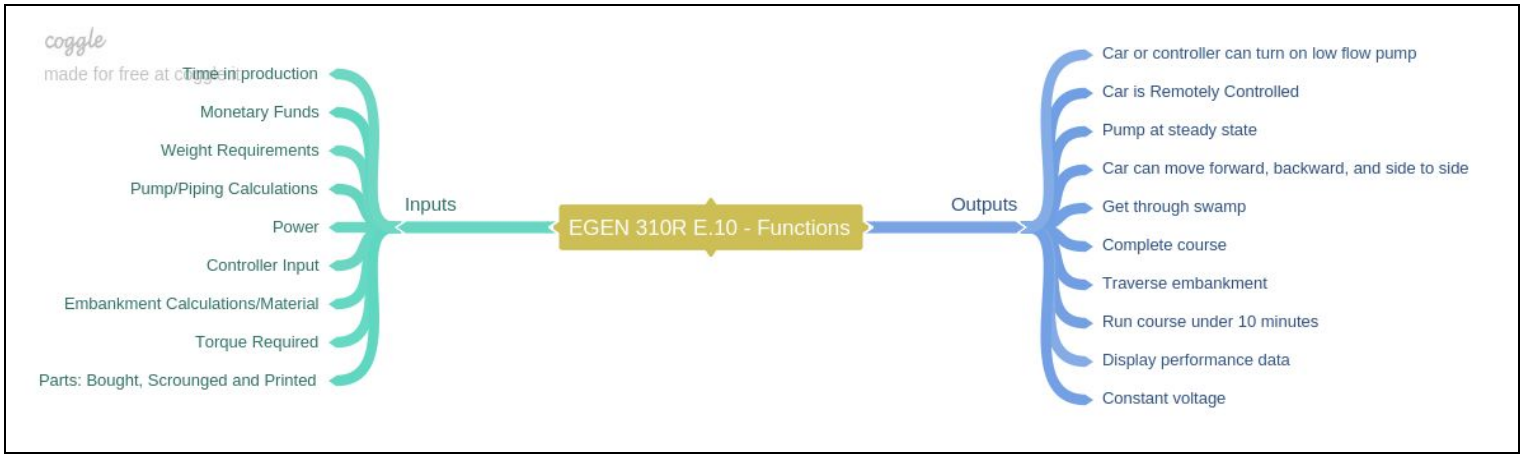

coggle
made for free at coggle.it
Time in production
Monetary Funds
Weight Requirements
Pump/Piping Calculations
Power
Controller Input
Embankment Calculations/Material
Torque Required
Parts: Bought, Scrounged and Printed
Inputs
EGEN 310R E.10 - Functions
Outputs
Car or controller can turn on low flow pump
Car is Remotely Controlled
Pump at steady state
Car can move forward, backward, and side to side
Get through swamp
Complete course
Traverse embankment
Run course under 10 minutes
Display performance data
Constant voltage

Figure 4. Green branches are the input tasks and blue branches are the outputs. The green is specifically what we need to do and the blue branches are how we're going to do it. The chart was made with Coggle, a free online flowchart designer.

By categorizing our inputs in this diagram our outputs became clear. The vehicle must be remote controlled, fast, handle a variety of movements, and travel our assigned elements.

In addition to the design phases our team had to meet the requirements of the Cat's Conundrum course. Physically our vehicle was constrained to a maximum size of 6"x6"x7.5" as outlined in the challenge. The design was further constrained by the manufacturing methods available. For fabricated parts, we were limited to the tooling of the Makerspace which made machining, cutting and bending custom metal parts. We were also given access to a 3D printing lab and of course whatever commercially available parts we could find within a budget of $250.

Other constraints included requirements for each major field. As a team, we are required to have a two foot section of "head loss" pipe to achieve steady state in the swamp, our embankment must be 2' x 2' and navigate a 90° corner, our control system must use a programmable chip and the vehicle must receive input from a graphical user interface, and the electronics must be composed of non-commercial parts with no change of power supply mid run.

Design Process Steps

To choose from our list of ideas we used a Morph Chart (Table 1) that allowed us to rank selections in a Pugh Chart (Table 2). The Morph Chart itself lists functions the car must perform. We received these functions from the Functional Diagram during initial design phases. The solutions listed in the Morph Chart are row independent. Solution selections are not dependent on which column they're in. The goal of this chart is to make a selection from each row that will be the best fit for the vehicle. The Pugh Chart examines various concepts and their combinations for viability.

Table 1. Morph Chart. Various solutions to functions are displayed.

Function	**Solutions**				
Display Data	Graph	Updating Value	Colored Status Indicator		
Controller	Click Buttons	Joysticks	Touchscreen	Keyboard	Gyroscopic
Communication	Wi-Fi	Bluetooth	Wired	Radio	
Climb Hills	Wheels	Tank Treads	Air Pistons		
Handle Bumps	Suspension	High ground clearance	Flipping self over		
Lift Heavy Gate	Heavy Vehicle	Wedge	High Speed Ram		
Pump at Steady State	Head loss to have flow rates equal	Switch that turns pump on	Size both pumps		
Create Head Loss	Mathematical Models	Test Different Lengths	Calculate Present Losses		
Embankment Structure	Iron	Wood	Foam		
Embankment Surface	Sand Paper	Foam	Cement	Wood	

This chart encompasses early design ideas to perform the various functions we've identified to complete. Our more creative options include using compressed air pistons for movement or using a phone's gyroscope and accelerometers to control the vehicle. Our idea to design a flippable vehicle was developed here. Without a live, onboard driver, it makes sense to design a vehicle to change its orientation to fit the environment. At first glance, constructing the embankment out of foam and using sand paper for the surface is attractive. It offers affordability, it's lightweight, and provides a high coefficient of friction to travel.

Table 2. Pugh Chart. Concepts are created from morph chart selections

Concepts - >	Weight	Concept 1	Concept 2	Concept 3	Concept 4	Concept 5
Objectives	0.7	0.35	0.49	0.28	0.56	0.14
Intuitive Controls	0.8	0.72	0.72	0.56	0.72	0.16
Inexpensive	0.4	0.12	0.24	0.08	0.28	0.08
Open weighted gate	0.5	0.45	0.45	0.25	0.45	0.1
Go up steep hills	0.6	0.18	0.24	0.12	0.36	0.12
Pump at steady state	0.6	0.54	0.48	0.42	0.54	0.12
Speed	0.2	0.1	0.08	0.04	0.16	0.04
Traverse embankment	0.6	0.48	0.3	0.18	0.12	0.12

The concepts found in the Pugh Chart are composed of selections from Table 1. The solutions are described in combination below.

Concept 1: This concept displays data to the driver via graphs. The vehicle is controlled by click buttons. The GUI will use Bluetooth communication. Wheels will be the method of travel so they will be coupled with suspension to handle bumps. To lift the gate the vehicle will be heavy. To pump at steady state we will use a packed bed based on mathematical models. The embankment structure will be made of iron with a surface of sandpaper.

Concept 2: The driver will receive data via an updating value and control the vehicle with joysticks. The GUI will use a bluetooth interface to control the car. We will climb hills with tank style treads and handle bumps with a suspension system. The vehicle will be heavy in order to push through the gate. Pumping at steady state will require use of 90° elbows for losses. The embankment structure will be composed of wood and the surface will be foam.

Concept 3: The driver will receive data via a colored status indicator and control the vehicle with a touch screen. Wireless communication will not be achieved and the car will have a tether. The drive system will use compressed air tanks to power pistons on a crankshaft. The air will be powerful enough to flip the vehicle to handle bumps and rough terrain. To lift the gate our vehicle will hold a wedge shape. To pump at steady state we will install a switch on the course for the vehicle to trip. The method of head loss will be baffles. The embankment structure will be composed of wood and the surface will be cement.

Concept 4: The driver will receive data via live, updating graphs and control the vehicle through keyboard inputs. The car will receive data over a Wi-Fi connection. To climb hills the vehicle will use wheels. To lift the gate the vehicle will ram it. To pump the swamp at steady state a packed bed will be used. A water level sensor, or float switch, will be used to turn on our pump. The embankment structure will be composed of foam and the driving surface will be wood.

Concept 5: The driver will receive data via a colored status indicator and control the car with click buttons on a visual display. The car will communicate with the GUI over a Wi-Fi connection. The drive system will use air compressed pistons and will handle bumps with a high ground clearance. The vehicle will lift the gate by ramming it. To pump at steady state we will use a packed bed. The embankment structure will be iron and the driving surface will be foam.

These concepts allowed us to examine various solutions in combination to see which components work best together. We learned that we all agree joystick based control would be the preferred control option. The tactile feedback is intuitive to a potential driver. With our flippable design, the vehicle would need to be light. So, in order to open the heavy gate we have to have a wedge-shaped body to prop the gate open while we travel through. For pumping at steady state, the packed bed is the easiest to model mathematically and we can perform testing once built. Much less time intensive that constant testing or attempting to construct baffles. The embankment will be easier to transport the lighter it is, so a foam structure is preferred. A surface of sandpaper/grip tape will be used on the element to maximize the amount of friction keeping our vehicle on the slope. After research, bluetooth was the most viable communication method. The modules are cheap and the straight pass through system is easily implemented.

Prototype Analysis

In Prototype 1 we had not yet integrated the electrical and mechanical systems together. We had a mechanical prototype of the frame and treads and another of the code and electronics running servos. From the mechanical prototype, we learned that the frame design we were using was solid, but that it put too much tension on the bolts and that it didn't have room to mount the servos as they were meant to be mounted. The vehicle was also too wide to easily fit through several of the obstacles. This led to an overall shrinking of the vehicle and electrical components. Due to the changes in the scale of the vehicle and components, mounting locations for the electrical components also had to change as the triangle mounting area enclosed by the tracks shrank significantly.

Prototype 2 integrated all systems and performed well on the Cat's Conundrum course. It was unable to complete the course within the allotted 10 minutes, but demonstrated the capability to climb ledges of approximately 2 inches and to navigate rough terrain obstacles. It was fully controllable via Bluetooth and was able to take advantage of its reorienting design by driving on all three faces. It also had the torque and traction to push through the medium weighted gate in the "Barge Through" challenge. It was unable to overcome the heaviest gate due to loss in traction though we feel we designed for enough torque to be able to accomplish the task on a different surface.

Major Points of System Integration and Potential Failure Modes

The major points of subsystem integration were the various interfaces: electronics mounting on the chassis, electrical switch for pump, the GUI's bluetooth connection to the onboard

Arduino, and the vehicle's ability to traverse the embankment. The footprint of the electronics was a large topic of discussion because it decided the ground clearance of the hypotenuse face of the treads. Should the electronics have been too large, our design wouldn't be able to viably change orientations smoothly. The switch for the pump needed to work as expected because steady state timing was critical. Major concerns with bluetooth integration were taken into consideration because if that system failed our vehicle would be unable to move. Should final design changes alter the center of mass our vehicle may have been unable to traverse the embankment at the desired angle also.

See Appendix A and the attached, zoomable PDF for the Failure Modes and Effects Analysis. Major concerns include jostling disrupting the electronics and creating an unsafe electrical connection, not enough torque to enter the weighted gate, and the overall durability of our components from the vehicle to the packed bed.

Project Management

Our management strategy was self enforced. Individual team members kept themselves on task in between deliverables. The pace of progress reports fit well with the design and build timelines. Our team had little difficulty finding time to complete our tasks. We made sure to discuss purchases as a team before making them and maintained clear and open communication over our group chat. Any discrepancies were cleared up while completing group reports. We couldn't have done much better.

Budget

This list of materials (table 1) is composed of all the components used in the final prototype. We're proud to report that our finished prototype was $100 under budget. We were able to make smart purchases that paid off in our design.

Table 1. Bill of Materials, out of pocket costs

Part #	Item Description/Supplier	Estimated Cost of Item	Actual Cost of Item	Quantity	Total
1	Bluetooth Module	$17.50	$9.99	1	$9.99
2	Microcontroller	$5.50	$11.65	1	$11.65
3	Servos	$25.90	$25.90	1	$25.90
4	Linear Regulator	$10.00	$7.57	1	$7.57
5	Batteries	$7.50	$4.48	2	$9.96
6	Battery Holders	$5.00	$4.52	1	$4.52
7	Custom PCB	$4.49	$4.49	1	$4.49

8	Wires/Connectors	$6.43	$6.43	1	$6.43
9	SMD Caps/Resistors/Fuses	$4.12	$4.12	1	$4.12
9	Foam for embankment	$10.99	$22.99	1	$22.99
10	Sand paper	$ 3.99	$3.99	1	$3.99
11	Adhesive	$ 2.50	$5.00	2	$10.00
12	Pump	$20.00	$8.99	1	$8.99
13	PVC Pipe	$10.00	$2.50	1	$2.50
14	#0-80 ⅜” Screws	$0.25	$0.27	8	$2.16
15	#6-32 1” Screws	$.10	$0.08	7	$0.56
16	#6 washers	$1.00	$1.15	36 count	$1.15
18	Track Kit	$8.07	$8.07	1	$8.07
15	#0-80 Nut	$0.25	$0.23	8	$1.84
16	#6-32 6” Threaded Rod	$1.00	$0.80	2	$1.60
17	#6-32 Lock Nut	$0.10	$0.09	4	$0.36
18	#6-32 Nut	$1.00	$1.15	24 count	$1.15
19	3D Printed Parts	$0.00	$0.00	11 prints	$0.00
	Total				$149.99
	Amount Remaining ($250-Total)				$100.01

Table 2. Bill of Materials, free or scrounged materials estimated costs.

Part #	Item Description/Supplier	Cost of Item	Quantity	Total
	3D printed fittings	$0.25	2	$0.50
	Pellets for packed Bed	$1.00	1	$1.00
			Total:	$1.50

Time on Task

Most team members completed their tasks on their own time. The only reasons to meet were for combined tasks like designing the switch for the pump, integrating the electronics onto the chassis, building the C code for the Arduino, and filming for Prototype 1. This being said, our team remained diligent and focused with a divide and conquer strategy. We all put in more than our fair share to complete the project.

Table 3. Time spent over the course of the semester on task

Student Name	Total Hours On Task for this reporting Period
Ivan Albert	64 hours
Michael Price	95 hours
Berke Gunaydin	63 hours
Ryan Benge	75 hours
Kyle Hagerman	100 hours
TOTAL	397 hours

Gantt Chart and Management Efforts

See Appendix B for our Gantt Chart and the attached pdf "Appendix B-1" to view it more closely.

Our management efforts were fairly independent. The team kept an active communication over a text group chat and emailed as necessary. Often we would discuss progress after lecture and held ourselves accountable for our contributions. The Gantt Chart served as a good reminder of our group's timeline though we only updated it on a submission basis. This being a somewhat small project we were able to effectively focus and complete tasks step by step without losing sight of the end goal. Should our project have been any larger and we could each devote more time to its completion I'm sure the Gantt chart would have been updated much more frequently. Without a project manager, it just wasn't a priority to update the schedule.

Vehicle Design

This vehicle was built with a multidisciplinary focus and thus each team member had specific tasks to complete within their area of expertise. Some design phases involved collaboration which our team was able to handle very well.

Structural Design

The structure of the vehicle was designed to be easily 3D printed, quickly assembled, and use few moving parts to reduce potential points of failure.
It used commercially available rubber tracks from a Tamiya Track and Wheel Set. The elasticity of the tracks allowed the tension of the tracks to be easily modified by changing the distance between wheels and rubber's high coefficient of friction provided much needed traction. Replacements for the wheels holding the tracks had to be printed due to the unavailability of axles that would fit them, so custom wheels were made and designed to fit freely rotating onto a #6 threaded rod.
The frame was 3D printed out of simple but sturdy parts. It consisted of two L shaped brackets with holes for the axles, and two platforms with built in cross braces which slotted into place. This allowed for a strong rigid structure to be made out of 4 easily printable parts and held together by 4 bolts. The servos were held in place by a 3D printed bracket which attached to the cross bar section of one of the platforms with 3 bolts.

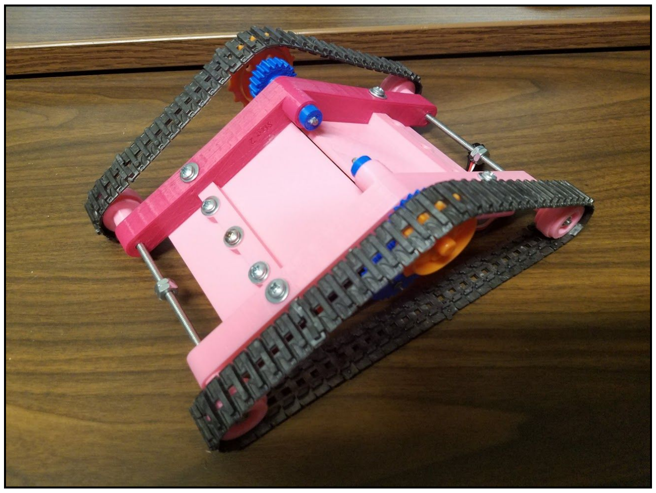


Figure 5. The wedge orientation shows the pieces and connections of the vehicle frame.

Electromechanical Design

The vehicle is driven by two independently controlled Pololu - SpringRC SM-S4303R continuous rotation servos. This allowed for a simple control scheme where both motors can be activated to drive forward or backward and by moving in opposite directions they allowed for a very tight turning radius. Due to the generally large size of standard motor-driver circuitry and very limited amperage throughput, DC motors which required this extra control circuitry were quickly eliminated as a design choice. Brushed DC motors are also known to produce significant electrical noise in circuits due to the inductive nature of their windings. The noise generated by the DC motors was observed during a test conducted on a potential DC motor design option. Photos of the observed noise are included in Appendix D as Figure D-2..

Considering the potential drawbacks of brushed DC motor option, continuous rotation servos were chosen as the intended motors for the tracks. Additionally, the servos provided more torque than the researched brushed DC motors in the same power draw range, and allowed for direct PWM control from the Arduino without an external motor controller. The needed torque was based on providing enough torque for the vehicle to lift itself and was initially calculated as

$$T = F * r * X$$

Where T is torque, F is the weight of the vehicle, r is the radius of the sprocket that drives the treads, and X is the factor of safety. At the time of purchase, it was estimated that the weight of the vehicle would be close to 1.27 lbs, the sprocket would have a radius of 0.75 in, and a safety factor of 2 was chosen. This gave the result

$$T = 1.905\ in - lbf$$

The SM-4303R servos provided 4.17 in-lbf of torque each, which was well above our estimate. As the design progressed a 3:2 gear ratio was also added between the motor and the sprocket which increased the torque further.

The motors' power was transmitted to the driving sprockets by way of 3D printed gears that interfaced with existing mounting hardware included with the motors. A lot of fine adjustment work was needed to make the gears mesh without skipping, but by Demo Day they worked reliably.

Control Circuit Design

A completed PCB circuit diagram with the ground fill mask removed (labeled Figure D-1) is provided in Appendix D for reference with this section. Below is a picture of the custom PCB resulting from the design process outlined in this section.

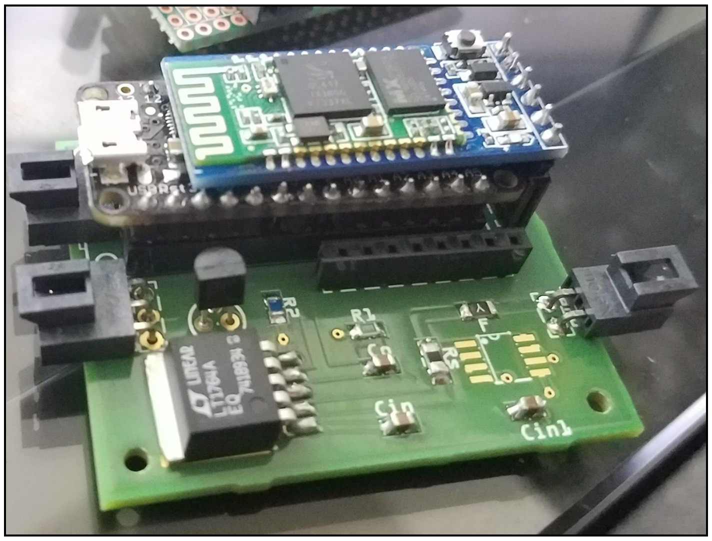


Figure 6. Assembled PCB For Prototype 2

Power for the electromechanical components was delivered from two standard 18350 form factor 3.7v rechargeable lithium-ion batteries in series. These batteries were rated at 1100mAh each, with a maximum instantaneous current draw of 10 Amps. With those ratings, Humphrey had roughly 26 minutes of runtime for an absolute worst case where both servos were at full stall with all control circuitry running. This resulted in nearly three hours of potential run time at light to moderate loading on the servos. Due to the standard form factor of these batteries, commercially available battery holders were available, which was ideal due to the lack of access to a spot welder for a holder-less design. Even considering the extra space of the battery holders added to each battery, a similarly mAh rated AA or AAA form factor battery pack would have been potentially three times the total physical volume, and still would not have had a sufficient maximum peak amp draw that would be safe for this vehicle design.

With both batteries in series, the voltage potential at nominal charge of the batteries was 7.4 volts, with a range of 3.0 to 4.2 volts each, or 6.0 to 8.4 volts across the full operating range. Because of this large range of operating voltage, and the danger of improperly regulated lithium ion batteries bursting into flames, regulation circuitry was carefully selected to avoid such an event. As tested, the potential full stall amperage draw of both servos and the control circuitry combined was roughly 2.4 Amps. Due to this potential high draw, power regulation for the components on Humphrey is provided by a custom PCB with a Linear Technology LT1764A adjustable LDO linear regulator coupled with filtering capacitors. This regulator features high transient response, .34v drop out, virtually no quiescent draw, integrated battery protection via on board circuitry, an enable/disable pin, thermal shutdown protection, and a controllable voltage output of 2.7v to 20v. These features made for an ideal linear voltage regulator for our battery powered car.

Linear regulators dissipate the excess power via heat, which results in potentially damaging temperatures during high amperage draw. To ensure the typical power draw through the regulator would not cause it to disable operation due to a thermal shutdown, several calculations based upon power drop through the regulator were performed to verify an addition of a heat sink or higher airflow would not be needed to maintain operation. The equation used was provided in the manufacturers documentation and was:

$$P_{Dissipated} = I_{OUT(MAX)} * (V_{IN(MAX)} - V_{OUT}) + I_{GND} * (V_{IN(MAX)})$$

This equation yields about 5.75 watts of power dissipated when the servos are at full stall, but the servos will likely give out at stall before significant heat was allowed to build in the regulator. Also, in the case where the servos are allowed to stay stalled that long, the regulator shutting down due to thermal protection may save the servos from breaking. Under normal operation, power dissipated is around .6 watts, which is well below the regulators free air dissipation capabilities, and will result in an ending temperature of only about 13 degrees Celsius above ambient temperature.

Additional supporting circuitry was also used on the PCB to set the regulators output voltage, and provide signal filtering. At the recommendation of the manufacturer, ceramic, low impedance capacitors with a value of 10μF were used. The signal filtering capacitors were also placed as close as possible to the inputs and outputs of the regulator to maximize their efficiency. To set the output voltage on the regulator, two resistors were used to provide a feedback loop to a sensing pin. The equation for the resistor ratio to output voltage is as follows:

$$V_{OUT} = 1.21V\left(1+\frac{R2}{R1}\right)+(I_{ADJ})(R2)$$

$$V_{ADJ} = 1.21V$$

$$I_{ADJ} = 3\mu A \text{ AT } 25°C$$

$$\text{OUTPUT RANGE} = 1.21V \text{ TO } 20V$$

The output voltage value of 5.95 volts was as close as could be obtained to the desired 6 volts with readily available resistors.

The servos had a maximum safe continuous operating voltage of 6 volts, so the voltage regulator was set at 6 volts, which required the battery side voltage would have to stay above the output voltage value added to the dropout voltage, or roughly 6.25 to 6.35 volts, to remain operational. This fit perfectly with the lithium-ion batteries, as one danger to the longevity of these types of batteries is drawing them below roughly 3 volts each, or 6 volts in series. With the combination of these design choices, a potentially damaging event could be avoided without extra considerations. Another danger to these types of batteries is heat due to malfunction, or drawing too much power. As these cells were unprotected, or "buttonless", a 5 amp SMD fast acting fuse was placed on the custom PCB before the regulator to cut power in case of multiple failures, or a significant shorting event. To help provide warning due to a cell or other circuit malfunction an analog temperature sensor was included next to the voltage regulator. This sensor would detect the surge in heat due to excessive load on the regulator or malfunction in the cell and alert the operator. While this sensor functioned and was included, we did not integrate it into this prototype phase.

At the heart of the PCB, and serving as the programmable microcontroller, was an Arduino Metro Mini Pro. This controller uses the same ATmega328 found on the larger Arduino Uno series, but was integrated into a much smaller package. The Arduino was chosen as the microcontroller for Humphrey because of its access to very powerful C language libraries, Michael and Kyle's experience with programming in C, its integrated FTDI controller, its size, lack of cumbersome operating systems, and access to plenty of PWM, analog, and digital I/O ports. The integrated FTDI controller helped keep cost down as a cable with the controller integrated would have cost extra. The Metro Mini Pro also had a built in voltage regulator capable of regulating 9 volts down to the required operating 5 volts, which worked well with the 6 volts coming from the LT1764A. The large amount of I/O ports also worked well with our initial design, which planned to have an accelerometer, current/power sensing IC, temperature sensor, PWM output for the servos, and the bluetooth device. As calculated, the Arduino was capable of supplying power to all of the peripheral devices via its built in regulator. In Prototype 2, the accelerometer and current/power sensing IC were not integrated, but were designed for on the PCB for later addition. These empty slots can be seen in Figure 6 above.

Due to the necessity to have the car controlled wirelessly, and the Electrical Engineers experience with the potential complications of paired RF communication, we chose to use a Bluetooth style transceiver. This allowed for seamless integration to most available control devices such as a phone or laptop with no additional circuitry. One of the benefits of the Bluetooth transceiver, is that the pairing process is very customizable, and very well controlled via the Bluetooth chips and their AT mode commands. Due to cost and size we chose to use the HC-05 model, which can function as both a master and slave device for extra flexibility with our end controller design. The Bluetooth transceiver also functions as a straight through device, which means that it does not add any additional code or require any decoding for the signals coming in or out of the RX and TX pins. These features were ideal as once it was set up, the device did not have to be changed in any way to allow full operation and still had the flexibility to connect to any other, non iPhone 8+, Bluetooth device.

Graphical User Interface for Control

To control Humphrey a graphical user interface, or GUI, was designed for use on a Windows based operating system. The Hooey program is described in more detail in Appendix E and in the attached document Appendix E-1. To fit with common programming practices and introduce strange names to the computing world the program is titled Hooey (Figure 5). It's a play on the abbreviation GUI with the 'H' from Humphrey. Hooey is coded in Python 3.4 and made heavy use of the Tkinter package. Tkinter is fairly simple to use and now with the TTK package the widgets match the standard operating system widgets to give your program a cleaner, more professional feel rather than being left with an outdated Windows XP look.

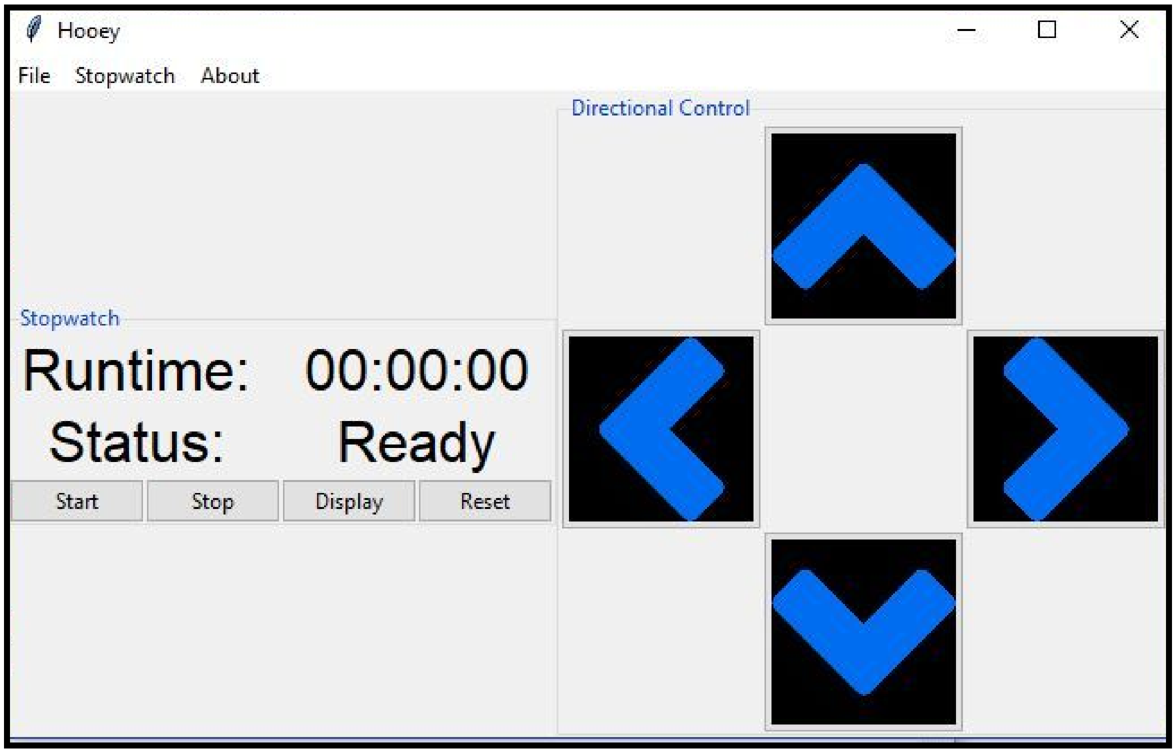


Figure 5. Hooey's layout. The Stopwatch frame is on the left and the Directional Control frame is on the right.

Upon opening Hooey, the user does not see a typical program window but rather the Python command prompt. After approximately 8 seconds the prompt will display the nearby bluetooth devices that your computer can pair with. Hooey connects via bluetooth to Humphrey once you select the correct integer value from the list of devices. Once a connection has been established, the program window will open.

The interface is divided into two frames. On the left, there is the stopwatch frame. There are various watch functions available including Start, Stop, Reset, and Display. The watch frame was included because the pump switch had an exact timing window to be tripped. By displaying current run time to the user, the driver knows when to trip the switch to set the swamp at steady state. Once the start button is pressed there is a display label that informs the user the clock is "Running". Ideally, the clock would automatically update itself every second. However, I could not figure out how to call the display_watch method from the main method. The GUI layer of coding adds complexity that I don't fully understand.

The right frame is the directional control frame. There are four buttons each holding an image of an arrow. They point up, down, left and right. They also do not have a function tied to them. Control of the car is actually input from the arrow keys of the keyboard with 's' for stop. The design was to have the buttons perform the same callback functions the keyboard did however I was unable to accomplish this. The only way I could get the bluetooth code blocks to run correctly was to put them outside the Hooey class. I don't know enough about calling methods from within a class instance of a window to be able to put the functions within the class.

With the Demo day deadline fast approaching the choice was made to simply get the car to function properly via bluetooth. So the code uses standard object oriented programming style to layout the GUI, but the bluetooth commands are all placed in the main method.

Figure 6 is a diagram of how the code works. Two devices are running a loop to listen for inputs. Whatever device is hosting Hooey runs the event loop that listens for keyboard inputs. When one of five keys is pressed then the method call sends data to the Arduino on board Humphrey. Humphrey's main loop is listening for a servo selection and then a setting either forward, stop or reverse. Both loops keep cycling until the vehicle and program are powered off.

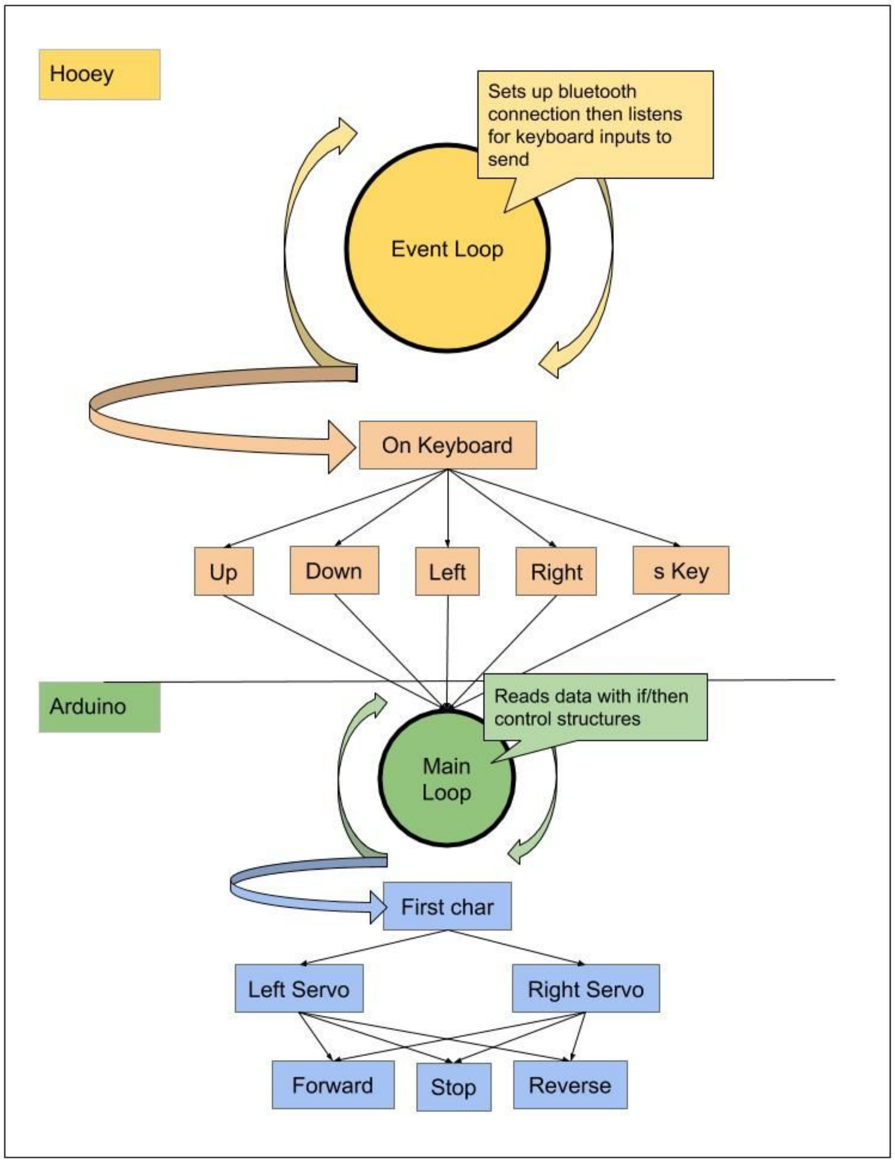

Hooey
Sets up bluetooth connection then listens for keyboard inputs to send
Event Loop
On Keyboard
Up
Down
Left
Right
s Key
Arduino
Reads data with if/then control structures
Main Loop
First char
Left Servo
Right Servo
Forward
Stop
Reverse

Figure 6. The UML diagram of the Hooey and Arduino code

The code onboard the Arduino is very simple. It initializes the bluetooth connection and waits to receive data packets from Hooey. Hooey sends 4 characters per command. The characters are two pairs. The first character in a pair selects a servo and the other character sets the state of that servo. All written code has been commented and is easily expandable.

Integration of Subsystems

Due to the inclusion of all of the control and peripheral circuitry on one compact PCB, the only electromechanical connections needed were from the servos and battery pack. To accomplish this, Molex brand Mini-Fit Jr's were used due to their rated 3 amp capabilities, compact size, polarized pins, and locking connections. This allowed the avoidance of any issues due to these connections overheating, rubbing, or coming loose during operation. All major components, with the exception of the voltage regulator, were also integrated onto the PCB with the ability to be swapped in case of malfunction during operation. The voltage regulator was directly soldered to the board to aid in thermal dissipation, minimize vertical space used, and ensure the issue observed where pins of the TO-220 package version of the same regulator came loose during operation.

Integration of the servos onto the frame was achieved via a 3D printed plate that held them in place to mesh with the gears turning the track system. The plate holding the servos also served as a place to attach the control circuitry board or batteries. Due to the decrease in form size, the batteries were attached on the plate area, whereas the control PCB was moved onto the vertical section. This orientation allowed for the best clearance possible with the current PCB design while Humphrey was flipped onto his long side.

Originally, the electronic circuits and batteries were going to be placed together in a 3D printed box and attached to Humphrey to provide protection from objects and liquids. Due to size and time constraints, this prototype was presented with the circuitry held tightly via zip ties with a mask of electrical tape to protect from accidental splashes of liquid.

Due to the use of a Bluetooth transceiver, integration of a controller to Humphrey was made simple. This is because Bluetooth is available in a broad range of devices including phones and laptops, which allowed for the desired controller to be programmed in any choice of environments and in any language desired. To integrate the controller to Humphrey, the Electrical Engineer verified function of the complete system, then passed the working code off to the Computer Science major for GUI integration. This path was the easiest and most time efficient as it cut out potential human communication issues, while still providing a base working system to branch off of.

Element Design

Our team was required to build or analyze two different elements in the course. Our Chemical Engineer, Ryan, sized the provided pump and purchased our own to cross the swamp element at steady state. Berke, the Civil Engineer, designed an embankment to use as a corner piece for the course.

Swamp

The swamp portion of Cat's Conundrum required the Chemical Engineering major to design a flow system to allowed Humphry to traverse the swamp via a buoyant bridge. In place was a high flow pump that was used to fill up the swamp using water from a nearby reservoir, and a low flow pump that was used to empty the water in the swamp back into the reservoir. The low flow pump, however, could not exceed half of the high flow pumps flow rate, so the designer had to model a way to create enough head loss in the high flow pump, to have the two flow rates be equal to one another.

Mathematical Model

In order to get these pumps to run at steady state, there had to be a pressure drop within the plumbing that would reduce the flow rate of the high flow pump enough for the two pumps to run at steady. The chemical engineering major on our team, Ryan Benge, decided that the best way to create this pressure drop would be to create a packed bed for the piping section. The reason he chose the packed bed is because he realized this would be the most effective route to creating enough head loss with only a two foot section of piping to work with. In order to accurately model this he used the Bernoulli equation (*Lumen*) and a variation of the Ergun equation (*Neutrium*) to model his packed bed. Some assumptions were made, which include: we assumed we were in a laminar regime, and we assumed the head loss due to the other fittings was negligible.

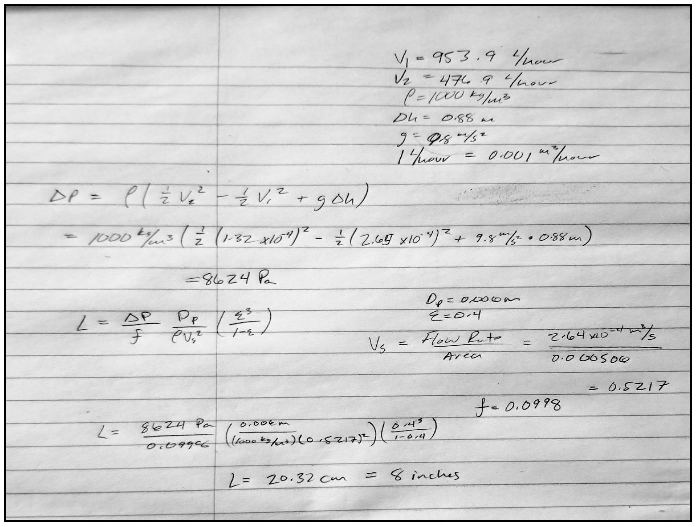
$$\textit{Bernoulli Equation}: P_1 + \tfrac{1}{2}\rho v_1^2 + \rho g h_1 = P_2 + \tfrac{1}{2}\rho v_2^2 + \rho g h_2$$

where:
- P is pressure
- ρ is density
- v is velocity of fluid flow
- g is the gravitational constant
- h is the elevation

$$\textit{Ergun Equation}: L = \frac{\Delta P}{f}\frac{D_p}{\rho V_s^2}\frac{\varepsilon^3}{1-\varepsilon}$$

where:
- Δ P is the pressure drop calculated from bernoulli equation
- D is the particle diameter
- ε is the void fraction (we use 0.4 because random packing)
- V is flow rate/area.
- f is the friction factor

In the modeling, the Bernoulli equation was rearranged in order to solve for a pressure drop (see figure 7). This calculated pressure drop was then put into the modified Ergun equation (with some known values) to calculate a bed length of around 8 inches.



$V_1 = 953.9$ L/hour

$V_2 = 476.9$ L/hour

$\rho = 1000$ kg/m^3

$\Delta h = 0.88$ m

$g = 9.8$ m/s^2

1 L/hour $= 0.001$ m^3/hour

$$\Delta P = \rho\left(\tfrac{1}{2}V_2^2 - \tfrac{1}{2}V_1^2 + g\Delta h\right)$$

$$= 1000\,\text{kg/m}^3\left(\tfrac{1}{2}(1.32\times10^{-4})^2 - \tfrac{1}{2}(2.69\times10^{-4})^2 + 9.8\,\text{m/s}^2 \cdot 0.88\,\text{m}\right)$$

$$= 8624\ \text{Pa}$$

$D_p = 0.006$ m

$\varepsilon = 0.4$

$$L = \frac{\Delta P}{f}\,\frac{D_p}{\rho V_s^2}\left(\frac{\varepsilon^3}{1-\varepsilon}\right)$$

$$V_s = \frac{\text{Flow Rate}}{\text{Area}} = \frac{2.64\times10^{-4}\,\text{m}^3/\text{s}}{0.000506}$$

$$= 0.5217$$

$f = 0.0998$

$$L = \frac{8624\ \text{Pa}}{0.09966}\left(\frac{0.006\,\text{m}}{(1000\,\text{kg/m}^3)(0.5217)^2}\right)\left(\frac{0.4^3}{1-0.4}\right)$$

$L = 20.32$ cm $= 8$ inches

Figure 7. Bernoulli Equation used for calculations

Vehicle Considerations

There really wasn't any design integration with the vehicle. The weight of the car was considered initially, because it was to rest on a buoyant bridge, however, we found that the amount of water displaced by the car was negligible due to the relatively low weight of the car.

Physical Design Descriptions

Pump:

The pump (figure 8) is a submersible pump that has a max flow rate of 3.5 gallons per minute. This pump was chosen because having it be submersible made it so there was less room for error while setting up the low flow pump. And the flow rate was below the flow rate of the high flow pump (so we thought). Unfortunately, the high flow pump didn't run at its max flow rate, rather it only ran at 4.2 gallons per minute. To counteract the higher flow rate of the low flow pump, tape was put over the pump reducing its max flow rate to 2.1 gallons per minute.

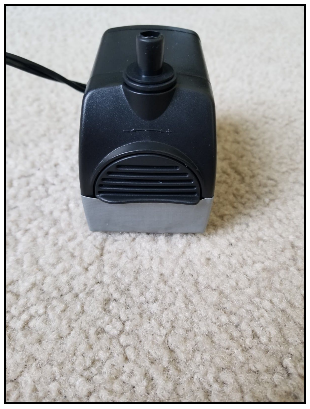


Figure 8: Pump with tape to reduce flow

Head Loss Section:
The head loss section was created by putting 6 millimeter pellets into a ½" section of PVC pipe. The pellets were held into place by 3D printed caps on the end of the pipe and inside of the pipe.

Switch:
Ryan and Michael worked together to make a switch (figure 9) that could be activated by the car. They chose a simple toggle switch that Michael was able to rewire to the pump. The switch was then mounted to a piece of wood, so that the car could run into it and activate the pump. We did run into problems with our run, though, due to the fact that the car wanted to climb the wood rather than just run into the switch.

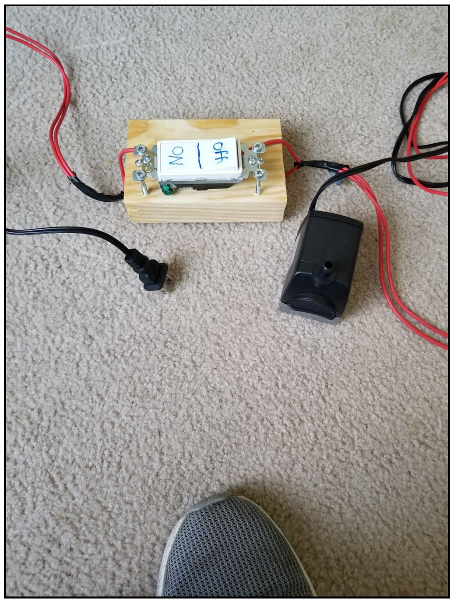


Figure 9. Switch with wiring to pump and plug

Validation of Mathematical Model
It seems that the modeling was pretty accurate given that the pumps were running at steady state, and the car was able to sit on the bridge for over a minute. However to get a more accurate model, the pumps would have to run together over a much greater time frame. Since we were working in such a short time frame, it is hard to determine if the pumps were actually at steady state, or just really close to steady state.

Embankment
I used foam to build the embankment (Figure 10). The embankment fit into 2x2 square ft area and it turns 90 degrees as per the design constraints. Sandpaper used as a surface to prevent the vehicle from sliding. The slope angle was 38 degrees. All calculations were performed before the CAD design model was built. When I finished my design I bought all materials from

Home Depot in bozeman. I shaved the foam at makerspace with hot wire cutter in the shape of bricks. The bricks were used to form the embankment (figure 11) and were cut into the triangles from mathematical calculations (figure 12).

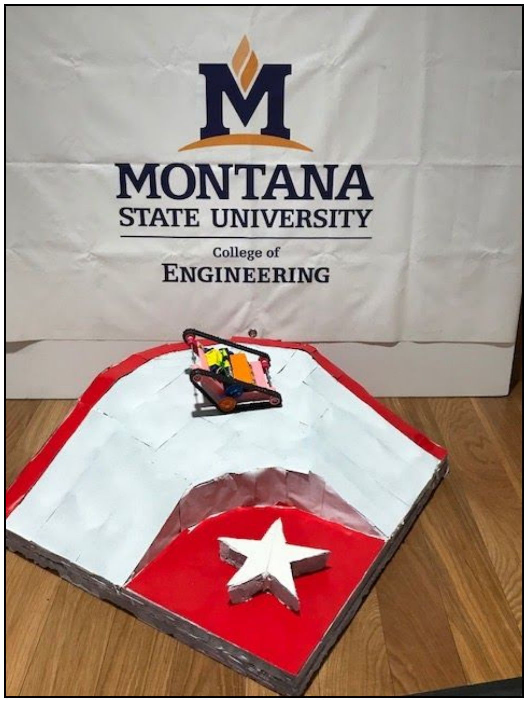
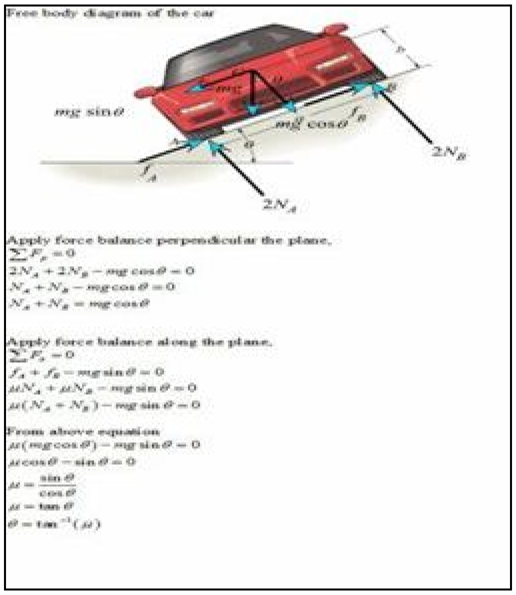


Figure 10. The completed Embankment with Humphrey

Mathematical Model

We used a website to calculate the angle between the vehicle and the embankment. Additionally, this website calculates the required frictional force required for the vehicle to not slide on the embankment. The coefficient of friction is related to the angle of the embankment modeled by basic physics equations (figure 13). If we increase the coefficient of friction we can increase the angle of the embankment as well. Preliminary sketches of the embankment are shown in figure 14. Further wireframe models can be found in Appendix F.



Apply force balance perpendicular the plane.

$\sum F_y = 0$

$2N_A + 2N_B - mg\cos\theta = 0$

$N_A + N_B - mg\cos\theta = 0$

$N_A + N_B = mg\cos\theta$

Apply force balance along the plane.

$\sum F_x = 0$

$f_A + f_B - mg\sin\theta = 0$

$\mu N_A + \mu N_B - mg\sin\theta = 0$

$\mu(N_A + N_B) - mg\sin\theta = 0$

From above equation

$\mu(mg\cos\theta) - mg\sin\theta = 0$

$\mu\cos\theta - \sin\theta = 0$

$\mu = \frac{\sin\theta}{\cos\theta}$

$\mu = \tan\theta$

$\theta = \tan^{-1}(\mu)$

Figure 13. A free body diagram of the car and relevant forces

N_A	*= Normal force applied from point A*
N_B	*= Normal force applied from point B*
f_A	*= Friction force at point A*
f_B	*= Friction force at point B*
μ	*= Friction force coefficient.*

Figure 14. Schematics of the embankment

Vehicle Considerations

To achieve the optimal path along the embankment, the width of the vehicle was considered as well as the material the treads were made of. With a rubber tread and a wide path for the vehicle to traverse these things were not issues. The velocity of the car was a concern as well, however the center of mass was kept very low to the ground so the vehicle did not tip until 55°. With the car's design being so optimal for traversing an embankment there was hardly any adjustments made to calculations.

Physical Design Descriptions

I designed my embankment to resemble a Turkish flag to represent where I'm from. The entire driving surface was covered in sandpaper and spray painted. The driving path was more than wide enough I did strength test and curve test before the run.

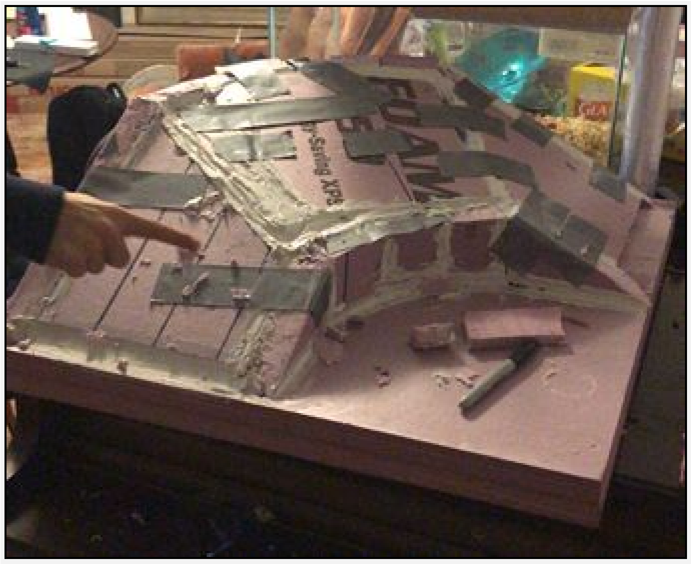


Figure 11. Embankment structure before surfacing

Validation of Mathematical Model

My mathematical model was totally correct. Our vehicle and embankment incorporated each other and the vehicle passed this track without any problem. For me our vehicle design is the best design for the cat course conundrum our vehicle passed all tracks easily. It passed the embankment easily too. We completed embankment perfectly.

Optimal Path Definition

The goal was to create the embankment so that the centerline of the curve was the optimal path. By adjusting the height of the outer edge of the embankment and the slope angle we were able to achieve this goal. Figure 15 shows the map of the optimal path over the embankment.

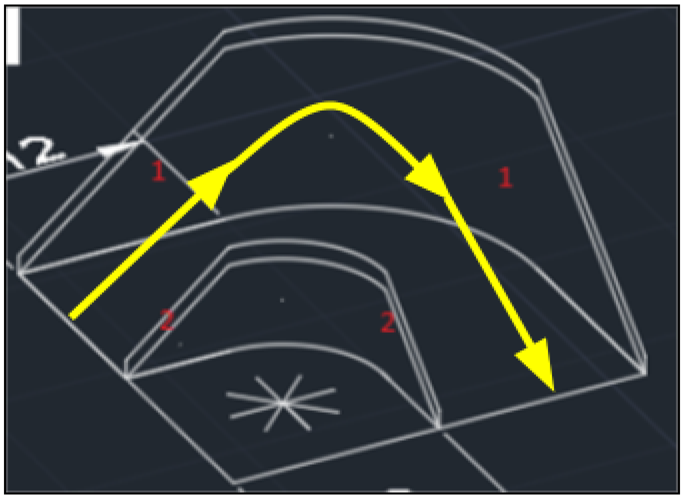


Figure 15. The optimal path for Humphrey to take is highlighted in yellow.

Mathematical Model

The embankment angle was chosen to be 38°. By equation x, the inner and outer edges of the embankment have to be certain lengths. the triangles 1 and 2 of figure 12 had to be certain lengths.

Equation x. $$arctan(4.74/6) = arctan(9.49/12) = 38°$$

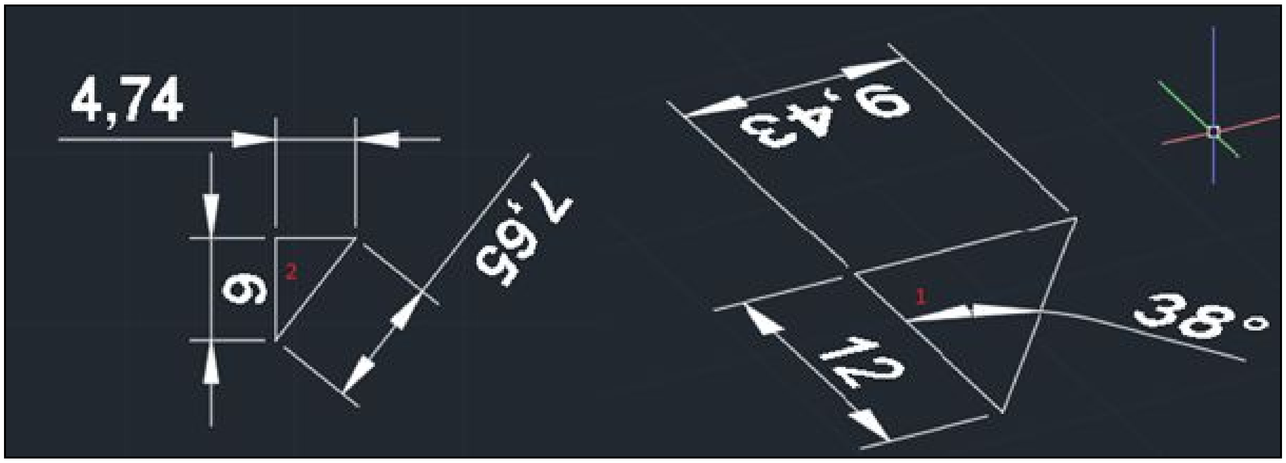


Figure 12. Triangle 2 is the inner side of the embankment and triangle 1 is the outer side. Measurements are in inches.

Conclusion: Team Self-Assessment

Our team tackled this challenge well. We used many design strategies learned in class to our advantage. Communication between team members also increased productivity. Given the pressure of deadlines and knowing that we were all working towards a common goal pushed us to complete our assigned tasks.

Teaming

Our final prototype was very functional. Humphrey was a successful example of multidisciplinary engineering done right. All the electronics integrated with the physical build and were easily controllable. We reached steady state at the swamp for the required length of time and the switch integration was simple. Not all of our designs became reality but we recognized when to cut our losses and make sure we had a functional prototype to demo come the deadline.

Project Management

Our project management strategy was fairly simple. We all held ourselves accountable for doing our own pieces to not let down the team. We were in communication about each major deadline and made sure to compile our work in advance before submissions to catch irregularities. Our team didn't have problems voicing concerns about various designs or deadlines. This allowed us to decide the best course of action as a group and everyone was always up to date on when tasks would be completed.

Design Goals

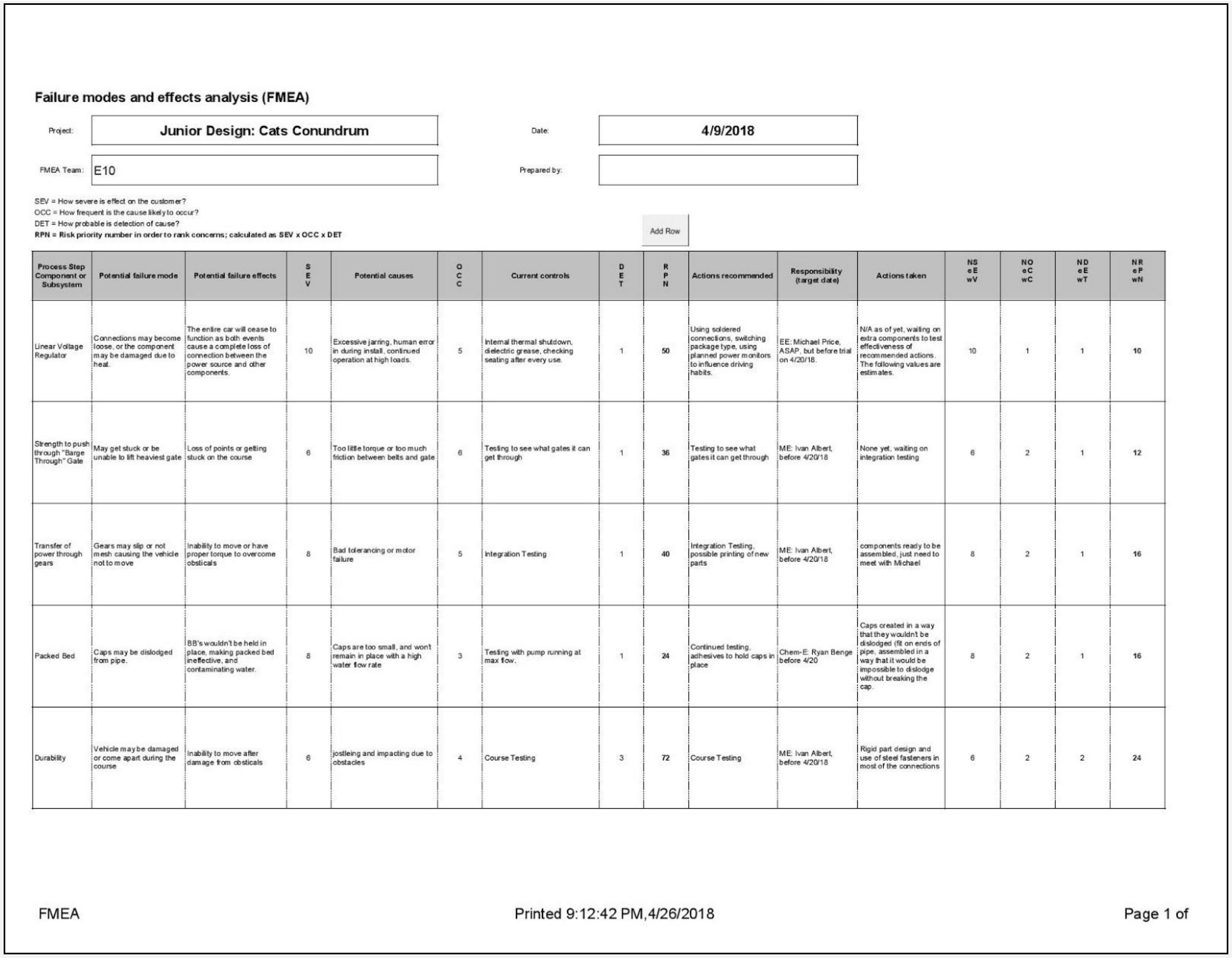
Our major design goals were to create a flippable vehicle that could drive on any treaded surface, open the weighted gate, and climb over rough terrain. We accomplished all of these things to some degree. Our vehicle could indeed drive on any surface though clearance became an issue for the hypotenuse-down orientation. We were able to lift the medium weighted gate because we didn't anticipate the loss of traction on the element surface and Humphrey could clamber over most obstacles given we could put the tracks on something solid. Given a Prototype 3 we would make sure to complete these goals. I'm sure we would alter the chassis to allow for greater ground clearance on all surfaces, we would perform further testing to open the heaviest gate and climb even steeper hills. We would also implement an earlier idea to produce a standalone controller run from a Raspberry Pi. The GUI would be hosted from a small tablet like device equipped with two joysticks to give the driver a more tactile driving experience instead of keyboard strokes. Also, missing from the GUI control was a way to switch the control directions for the different orientations. Given more time, we would install some kind of sensor to read the angle the vehicle was at to alter the controls so that forwards always drove the vehicle forwards. Due to the clearance issues of the electronics, they would be redesigned to take advantage of the surface area of each plate rather than vertical space.

Acknowledgements

As a team we would like to thank Mrs. Varnes for all her help and guidance. She made sure to lead us towards the direction of success. She was always very encouraging and understanding of our position as students. We owe a lot of our success to her teachings.

A great big thank you goes out to all the staff of the MSU Makerspace especially Matt Griffin. We appreciate the hours they stayed open for all students to come in for help and guidance. The workshops held in the space were also quite helpful. We're looking forward to seeing the expansion in the new hall next year!

Appendix A: FMEA



Failure modes and effects analysis (FMEA)

Project:	Junior Design: Cats Conundrum	Date:	4/9/2018
FMEA Team:	E10	Prepared by:	

SEV = How severe is effect on the customer?
OCC = How frequent is the cause likely to occur?
DET = How probable is detection of cause?
RPN = Risk priority number in order to rank concerns; calculated as SEV x OCC x DET

Add Row

Process Step Component or Subsystem	Potential failure mode	Potential failure effects	SEV	Potential causes	OCC	Current controls	DET	RPN	Actions recommended	Responsibility (target date)	Actions taken	NS eE wV	NO eC wC	ND eE wT	NR eP wN
Linear Voltage Regulator	Connections may become loose, or the component may be damaged due to heat.	The entire car will cease to function as both events cause a complete loss of connection between the power source and other components.	10	Excessive jarring, human error in during install, continued operation at high loads.	5	Internal thermal shutdown, dielectric grease, checking seating after every use.	1	50	Using soldered connections, switching package type, using planned power monitors to influence driving habits.	EE: Michael Price, ASAP, but before trial on 4/20/18.	N/A as of yet, waiting on extra components to test effectiveness of recommended actions. The following values are estimates.	10	1	1	10
Strength to push through "Barge Through" Gate	May get stuck or be unable to lift heaviest gate	Loss of points or getting stuck on the course	6	Too little torque or too much friction between belts and gate	6	Testing to see what gates it can get through	1	36	Testing to see what gates it can get through	ME: Ivan Albert, before 4/20/18	None yet, waiting on integration testing	6	2	1	12
Transfer of power through gears	Gears may slip or not mesh causing the vehicle not to move	Inability to move or have proper torque to overcome obsticals	8	Bad tolerancing or motor failure	5	Integration Testing	1	40	Integration Testing, possible printing of new parts	ME: Ivan Albert, before 4/20/18	components ready to be assembled, just need to meet with Michael	8	2	1	16
Packed Bed	Caps may be dislodged from pipe.	BB's wouldn't be held in place, making packed bed ineffective, and contaminating water.	8	Caps are too small, and won't remain in place with a high water flow rate	3	Testing with pump running at max flow.	1	24	Continued testing, adhesives to hold caps in place	Chem-E: Ryan Benge before 4/20	Caps created in a way that they wouldn't be dislodged (fit on ends of pipe, assembled in a way that it would be impossible to dislodge without breaking the cap.	8	2	1	16
Durability	Vehicle may be damaged or come apart during the course	Inability to move after damage from obsticals	6	jostleing and impacting due to obstacles	4	Course Testing	3	72	Course Testing	ME: Ivan Albert, before 4/20/18	Rigid part design and use of steel fasteners in most of the connections	6	2	2	24

FMEA Printed 9:12:42 PM,4/26/2018 Page 1 of

Figure A-1. Our FMEA chart. Examined systems include the mechanical chassis, the integrated electronics, and the packed bed for the steady state. See the attached pdf titled "Appendix A-1" to read more clearly.

Appendix B: Gantt Chart

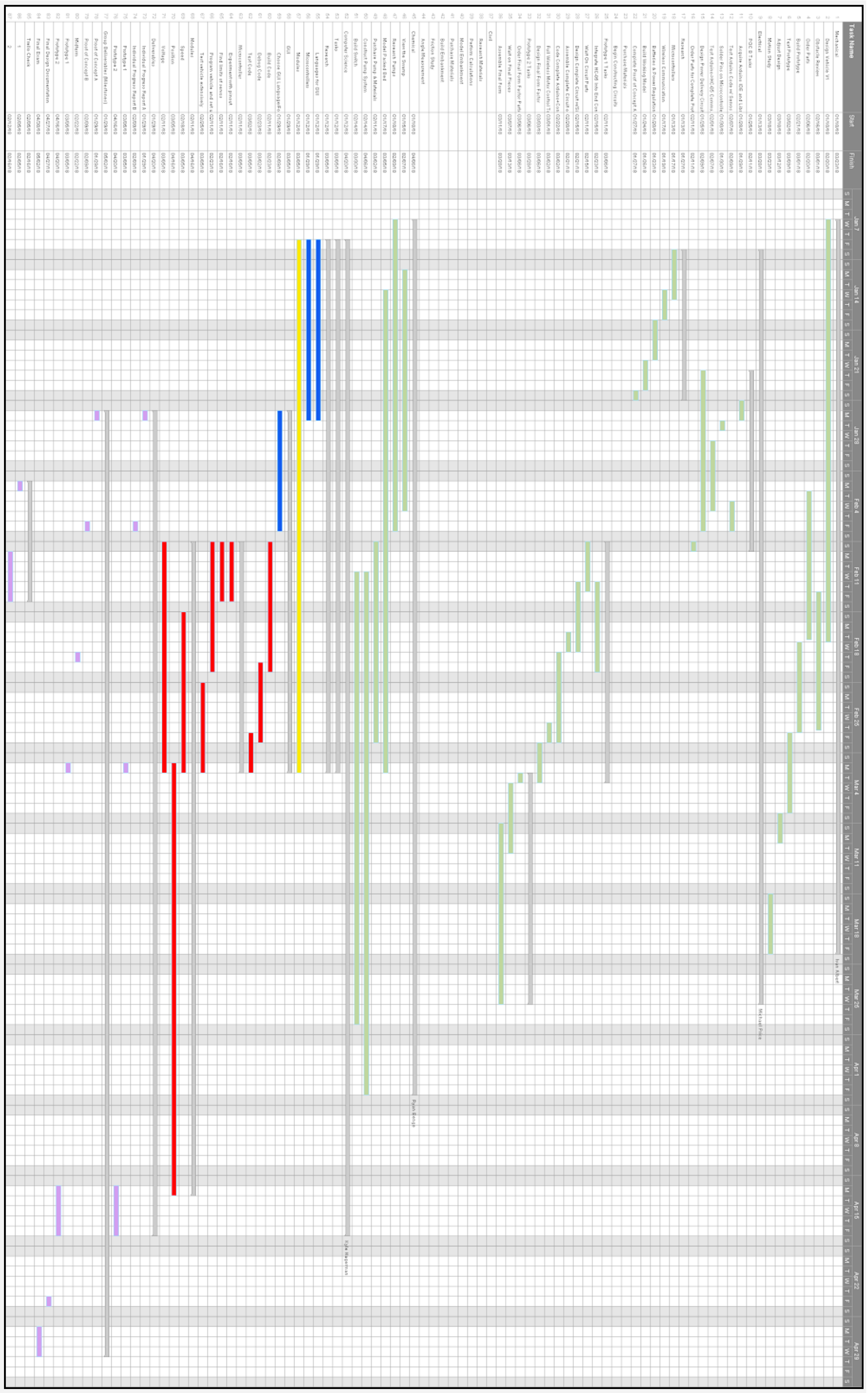


Figure B-1. The Gantt chart for Humphrey. Please see the attached pdf titled “Appendix B-1” to view more clearly.

Appendix C: Mechanical Component Drawing Package

Attached to this document are 11 pages of parts details. Please refer to these drawings for technical specifications.

Appendix D: Printed Circuit Board Design and DC Motor Waveform

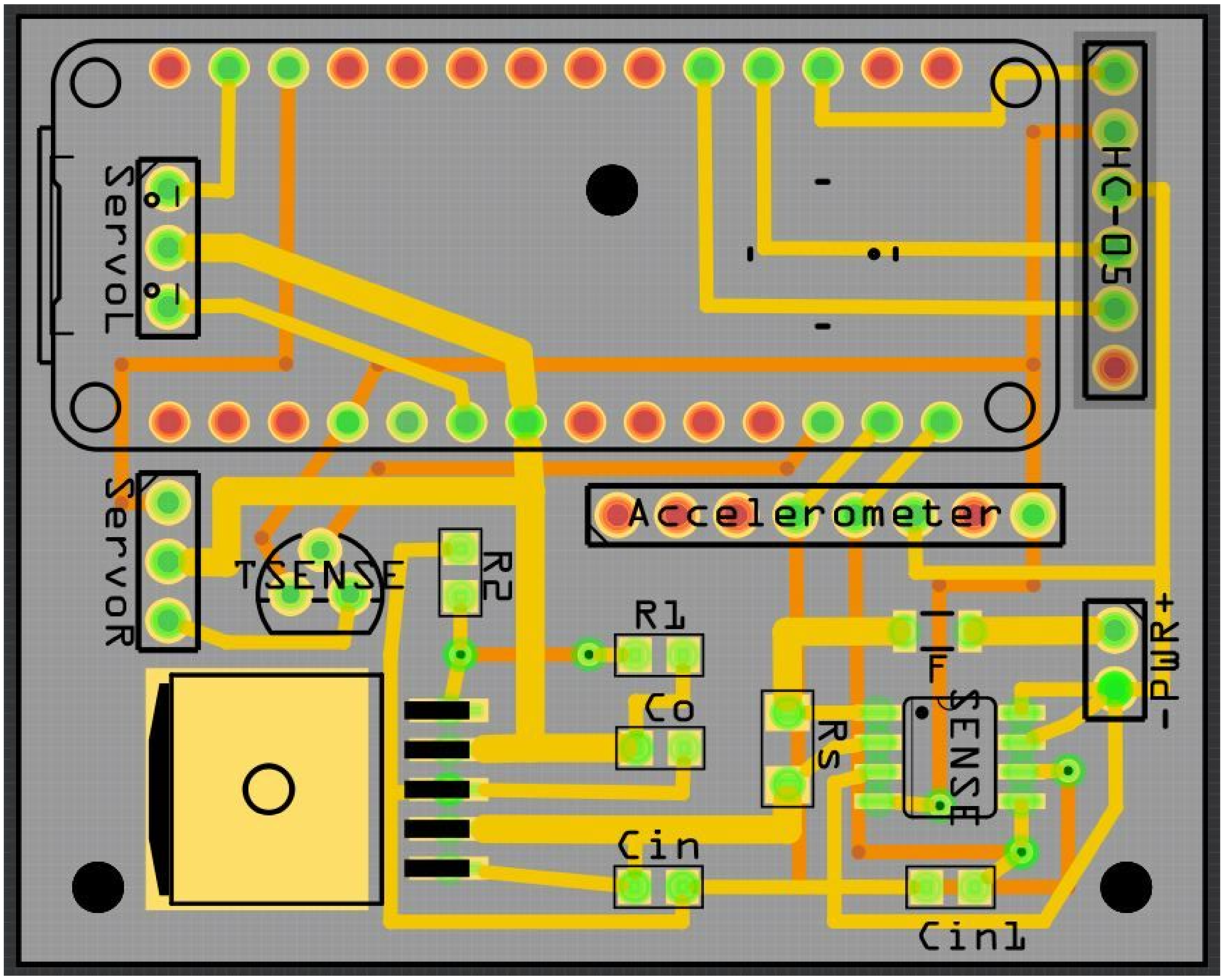


Figure D-1. PCB Design

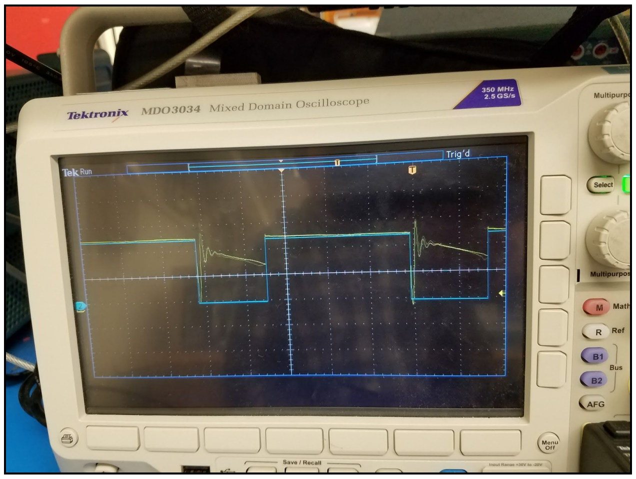


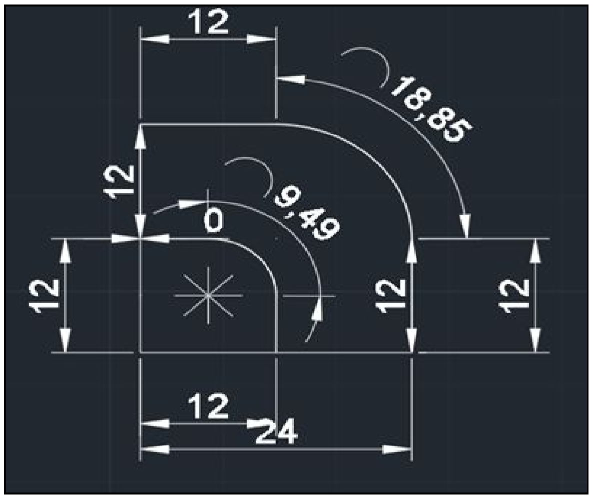
Figure D-2. DC Motor Noise

Appendix E: Code for Graphical User Interface and Onboard Arduino

Two bodies of code are included in this project. The first is written in Python and was compiled with Python 3.4. This version was used to allow for the integration of the PyBluez package because it's only compatible up to this version. Please see the attached .txt file titled "Appendix E-1" for the Hooey GUI code. It is well commented and fairly straightforward.

The second body of code is written in C for the Arduino. Included libraries are Servo.h and SoftwareSerial.h. There are basic variable declarations in the beginning of the file as well as servo initializations and the pinouts are set. The main loop just accepts input one character at a time from Hooey and sets motor states to forward, stop or reverse. Note that because it doesn't process all incoming characters at once, keys pressed in Hooey need to be held for at least one second or until Humphrey responds with the directed command. Delay has been minimal but noticeable. The data is only sent while the keys are being held down, once the key is released then Humphrey will continue to perform his last directed command. Please see the attached .txt file titled "Appendix E-2" for the C code. It is well commented and a very short file.

Appendix F: CAD Embankment Wireframe Model



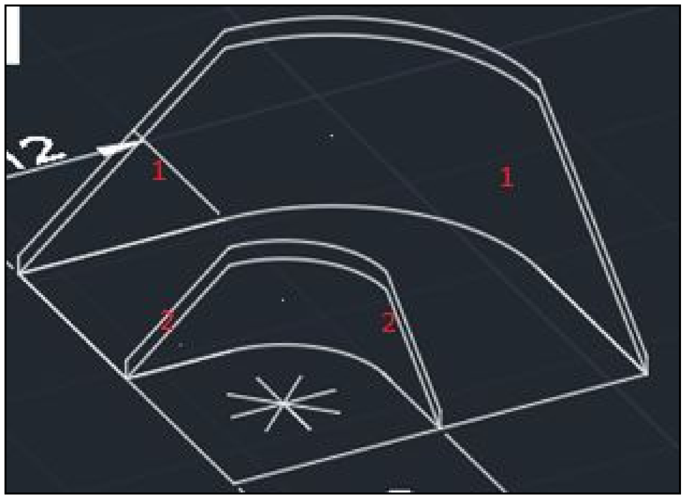


Figure F-1, F-2. Schematics of the final embankment design

References

"Bernoulli's Equation." *Lumen*, Lumen Learning, courses.lumenlearning.com/physics/chapter/12-2-bernoullis-equation/.

"Documentation." *Arduino - Docs*, www.arduino.cc/en/Main/Docs.

Hanly, Jeri R., and Elliot B. Koffman. *Problem Solving and Program Design in C*. 8th ed., Pearson, 2015.

"How to Download and Install Python Packages and Modules with Pip." *YouTube*, YouTube, 21 Jan. 2015, www.youtube.com/watch?v=jnpC_Ib_lbc.

"Less Talk, More Action." *Smartsheet*, www.smartsheet.com/.

Linear Technology, *Datasheet For LT1764A*. 2002. Linear Technology.

Lutz, Mark. *Programming Python*. 4th ed., O'Reilly, 2011.

"Pressure Drop through a Packed Bed." *Pressure Drop Through a Packed Bed – Neutrium*, Neutrium, 23 July 2013, neutrium.net/fluid_flow/pressure-drop-through-a-packed-bed/.

"PyBluez." *PyPI*, pypi.org/project/PyBluez/.

"Python 3.4.8 Documentation." *Overview - Python 3.4.8 Documentation*, docs.python.org/3.4/.

Stone, Barron. "Python GUI Development with Tkinter." *Lynda.com - from LinkedIn*, Lynda.com, 30 July 2014, www.lynda.com/Tkinter-tutorials/Python-GUI-Development-Tkinter/163607-2.html.