

# Wall Following Robot

A Major Qualifying Project Report:

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Benjamin Dwyer

April 25, 2013

Advised by

Professor Emanuel

# Abstract

This project explores the development of an educational robotics platform that focuses analog circuit design and models the specific behavior of a wall-following controls loop under varying conditions. While most platforms encourage mechanical development, this platform allows users to rearrange magnetically interconnected circuit components. Each component is current-protected and displays the signal travelling through it, thus users can observe as the behavior of the device changes and begin to develop a qualitative understanding of analog circuit design. This implementation of this platform follows walls. The task is simple to implement, yet the control theory behind the negative feedback is complex enough that modifying the circuit can produce some very interesting results.

# Contents

Abstract.....	1
Table of Figures .....	3
Executive Summary .....	4
Introduction to the Problem .....	5
What am I Actually Trying to Do? .....	7
Chapter 1 - Background Material .....	8
Robotics .....	8
Lego Mindstorms .....	9
Vex Robotics .....	12
The Boe-Bot.....	14
Electronics Kits .....	15
130 in One Kits.....	15
Snap circuits.....	16
Chapter 2 - Theory .....	18
Core component choices .....	18
Distance Sensor .....	18
Motor .....	21
Battery.....	21
Overview of each circuit block.....	22
Triangle Wave Generator.....	23
Sensor Block.....	24
Obstacle Avoidance .....	25
Right Wheel PWM block.....	26
Left Wheel PWM block.....	27
Controls Loop.....	27
Deriving the position equations.....	28
PSPICE Control Simulations .....	29
Chapter 3 - Alpha Implementation.....	31
Mechanical Design.....	31
Electrical implementation .....	32
Oscillator .....	33

Wall Follower Offset.....	33
Obstacle Avoidance .....	35
PWM Generator.....	35
Results.....	36
Chapter 4 - Results and Conclusion .....	37
Appendix A: PSpice Code .....	38
Wall Following Control Loop Simulation: .....	38
Overall Circuit Simulation:.....	39

## Table of Figures

Figure 0-1: K'Nex Car .....	5
Figure 1-1: Inside the Lego Mindstorms NXT Brick .....	8
Figure 1-2: Basic Mindstorms Robot .....	9
Figure 1-3: Lego NXT Brick .....	10
Figure 1-4: Mindstorms Light Sensor.....	10
Figure 1-5: Programming the Mindstorms NXT .....	11
Figure 1-6: Vex IQ System.....	12
Figure 1-7: Vex IQ Robot Brain .....	13
Figure 1-8: VexIQ Gyroscope Sensor.....	13
Figure 1-9: Parallax Boe-Bot.....	14
Figure 1-10: BASIC Stamp Microcontroller .....	14
Figure 1-11:130-in-1 Electronics Kit.....	16
Figure 1-12: Snap Circuits .....	17
Figure 2-1: Parallax Ping))) .....	19
Figure 2-2: Sharp GP2Y0A21YK0F .....	19
Figure 2-3: Operating Spec for Distance Sensor .....	20
Figure 2-4: Real World Test of Distance Sensor.....	20
Figure 2-5: Distance Sensor Output.....	20
Figure 2-6: The Motor .....	21
Figure 2-7: Block Diagram .....	22
Figure 2-8: Triangle Wave Generator .....	23
Figure 2-9: Oscillator Simulation .....	23
Figure 2-10: Summing Amplifier .....	24
Figure 2-11: PSpice Sensor Simulation.....	25
Figure 2-12: Obstacle Avoidance .....	26
Figure 2-13: Obstacle Avoidance and Wall Following .....	26

Figure 2-15: Control Loop .....	27
Figure 2-16: Robot at Origin .....	28
Figure 2-17: Position Tracking.....	30
Figure 2-18: Increased Feedback .....	30
Figure 3-1: Alpha Chassis .....	31
Figure 3-2: The First Prototype.....	32
Figure 3-3: Circuit Board.....	32
Figure 3-4: Oscillator Output.....	33
Figure 3-5: Distance Sensor as Distance Changes.....	34
Figure 3-6: Distance sensor as bias changes .....	34
Figure 3-7: Motor Driver .....	35
Figure 3-8: PWM at Minimum .....	36
Figure 3-9: PWM at Maximum.....	36

## Executive Summary

This MQP poses an idea for an educational robotics platform that focuses solely on electrical innovation. The robot is analog and its behavior is entirely controlled by rearranging circuit elements. Ideally each circuit element will be protected from overcurrent conditions and will display the signal travelling through it. By allowing young students to freely rearrange components without fear of damaging them or hurting themselves, changes in the circuit will be evident as changes in the behavior of the robot. Such a system will allow young children to build an intuitive, effortless understanding of how electrical systems actually work.

In this MQP, I first look at the work done by existing robotics platforms, including Lego Mindstorms, the Vex IQ system, and the Boe-Bot, as well as electronics prototyping systems such as Snap Circuits, and analyze where they have succeeded and failed. I then go on to design a simple analog wall-following robot. I then analyze the behavior of the positional feedback loop, derive equations that can be used to model the position of the robot in space, and simulate the behavior of such a robot in PSpice.

I then built, tested, and carefully analyzed the performance of this wall-following robot. This lays the framework for the rest of the robotics development platform while thoroughly exploring the behavior of a wall-following robot.

## Introduction to the Problem

At a fairly early age, children develop a strong curiosity for the world around them. I remember that in my own childhood, I was fascinated learning about levers, forces, gravity, vacuum cleaners, combustion motors, and everything else around me. I, like so many other children, had a tendency to take things apart and try put them back together. This works very well for mechanical devices--watching gears turn gives you an idea of how they actually work. Watching how some parts of a system turn faster and slower gives you a starting place for looking at the difference between torque and speed. Tiny plastic parts lend themselves well to being rearranged. After watching me spend hours fascinated by the mechanics of my model car elevator, my parent started getting me toys that allowed me to build my own mechanical contraptions.



*Figure 0-1: K'Nex Car*

There are all sorts of toys out there that encourage children to explore mechanical physics. K'Nex provides you with battery packs that have switches and attach to motors. You can attach gears to the motors and build all sorts of interesting mechanical systems. Lego Mindstorms simpler kits provide a basic microcontroller and some sensors that can actuate different motors. This allows children to build quite complex mechanical devices. Lego Mindstorms NXT platform and the Vex Robotics platform introduce a new factor--a programmable microcontroller. This allows young people to build very complex robots that can perform basic

autonomous functions based on drag and drop and actual C programming interfaces. There's a missing link in these systems though; between the code on the screen and the movement of the robot there is something inexplicable happening.

Taking apart my Lego Mindstorms controller revealed a world of components too fragile and complex for me to play with and understand. As I got older I spent hours poring over descriptions of computer architecture, reading about RAM, clock speeds, bus speeds, and computer architectures, but this never really answered my question. What happens between the code on the computer and the moving motor? What do all those tiny black chips do? All my reading did was introduce new questions. What was a transistor? Was it a switch? Was it an "amplifier"? What did it actually do? What was voltage, except something that came out of a battery? What on earth was current? Why did some parts get hotter than others? It took nearly 10 years before I could take my first class in electrical engineering and begin to develop an intuitive feeling of what a component actually does.

The information is spectacularly difficult to find and understand. Electrical Engineering is something that is taught in college, "normal" people just need to know that they push the "quick cook" button on their microwave and their food gets hot--unless it's metal, metal is bad in microwaves. Outside of college classes and engineering degrees there is no way for anyone, adults or children, to gain a basic, intuitive understanding of electrical concepts.

There are a few kits out there that try to approach this. I remember one, little springs connected components to each other, and if you followed the simplified wiring diagrams, you could make it produce motorcycle noises, or light up LEDs. But the first time I tried to make something without explicit guidance, I blew up one of the three transistors and the board never worked again. There was no coherence to this device though, as much as I followed their simplified diagrams, I never made the connection to the circuit schematics. I didn't understand what component values were, or what to do and what not to do.

That's the crux of the issue. There is currently no way for children--or adults, or even electrical engineering students--to easily develop a strong intuitive grasp of the basic principles of

electrical engineering. What is a transistor? To understand that you need multi-dimensional calculus and grad school.

This is the issue I am trying to approach. I want to build a kit that allows children to play with circuit elements, to exchange components for others, to watch how signals flow, and to produce some sort of concrete change in a real world system based on modifications of a circuit.

### What am I Actually Trying to Do?

Fundamentally, this project isn't about robots at all, it is about playing with circuits. The robot is only a tool I am using to make the effects of a circuit produce an interesting interaction with the physical world. Thus, the most core of the project is developing the component packaging—a package which makes circuit components easily connectable without any risk of damage, while also clearly showing signals as they pass from one component to the next

Once the packaging for components is completed, components will have to be assembled into circuits that make the robot do interesting things. This is where things start to get really fun. I need to take these repackaged components and use them to make beautiful, easily understandable circuits that effectively program the behavior of the robot. Not only do I need to make the robot work, I need to find several ways that the robot can work. I need to provide different feedback loops, different driver circuits, and different ways of processing signals in order to show that there are a near infinite number of ways to achieve a task.

When I can successfully communicate that there is always room for improvement, always different designs, and always pertinent and interesting theory behind them, I will have laid the groundwork for other people to experiment.

This is not just another robotics kit, this is a tool intended to get people interested in looking at circuits and understanding how they work, how they can be modified, and how they can be improved. It is a tool that attempts to demystify electrical engineering.



## Chapter 1 - Background Material

Before attempting to design this robotics platform, we will first take a look at both existing robotics and electrical development platforms. We are primarily interested in analyzing the functionalities of each platform. This includes studying how people respond to them, looking at what their functionalities are, and exploring what they actually do. We are also interested in looking at the target markets and pricing for these various platforms. As the product we are developing competes indirectly with all of them, it is very useful to understand where each has succeeded and failed.

### Robotics

Most existing robotics kits focus on mechanical innovation. They allow students to actually “build robots,” and by that they mean that can build the physical shell of the robot. Most these days also allow you to somehow program the robot itself, changing its behaviors and what it actually does with the mechanics you’ve built. The issue with all these is that the controller itself is a black box. All of the electrical aspects of the robot, the magic that change lines of code on the computer into motions of your structure, is completely inaccessible. You simply aren’t meant to open it. And if one dares to break the dreaded “no user serviceable parts inside” label and void the warranty on that fragile, expensive box, nothing inside makes sense anyway. You need a degree in microelectronics to be able to actually understand everything going on under the Atmel logo.



*Figure 1-1: Inside the Lego Mindstorms NXT Brick*

Part of the reason robotics development platforms are successful is because you can very easily demonstrate mechanical concepts, like center of gravity, to young children: “when this heavy object goes past this point, the robot falls over. Unless you hold it down, or add a counterweight, or...” You can simply *show* them and let them play with it. Microprocessors, though are not so easy. Try explaining a barrel shifter, program counter, shift registers, bus interfaces, and everything else going on inside a simple RISC microprocessor to anyone—it’s not easy to teach a college class, let alone a child. The technology is difficult to build, hard to demonstrate, and impossible to reverse engineer without an electron microscope.

Demonstrating concepts like the center of gravity of an object works well because a student can make a change and see an effect. This is very easy to understand. Changing one variable affects the output in a certain way. This is why educational robotics can be such a fantastic teaching tool. Changes you make in the software and hardware produce immediately observable outputs.

### Lego Mindstorms

Lego Mindstorms is an entire line of products, but their most recent robotics kit is the “NXT 2.0”. This \$275 kit (Mindstorms) is targeted toward students between 10 and 18 years old, and includes the NXT Intelligent Brick, an ultrasonic rangefinder, a color sensor, two touch sensors, three servo motors, cables, software, and over 600 construction pieces. Also included are instructions that allow users to build several different robots. Robots can be two-legged, have wheels, tank treads, or be completely stationary.



*Figure 1-2: Basic Mindstorms Robot*

At the heart of every robot is the \$150 NXT Intelligent Brick. The brick contains a 32-bit Arm-7 microprocessor running at 48MHz and has four input ports for sensors and three output ports for servo motors. The brick communicates via a USB cable or wirelessly via Bluetooth, and each is powered by 6 alkaline AA batteries. More than one brick may be used on a robot if more I/O capability is needed. More than one brick may be used on a robot if more I/O capability is needed.



*Figure 1-3: Lego NXT Brick*

Their product lineup is surprisingly expensive, for example their light sensor costs \$20 (Mindstorms) for pretty plastic packaging over a 10 cent component. While this may expanding the original kit prohibitively expensive, the professional packaging ensures that components cannot be easily damaged.



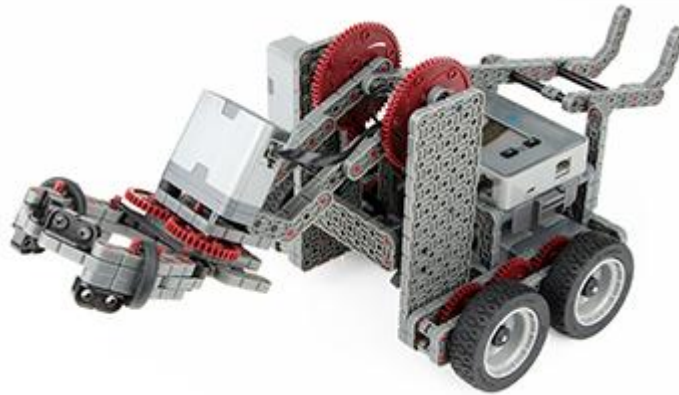
*Figure 1-4: Mindstorms Light Sensor*

The software supports a drag-and-drop style of programming. This is an attempt at making embedded programming simpler for younger children, but parents complained that “programs become cluttered quickly... the ‘language’ is stilted and many functions just don’t work.” (Amazon, sppooo). Another parent complained that while it is a fairly flexible platform, it “needs further parts to make it really useful” (amazon: Steve E. Jackson "Washad"). Almost all



## Vex Robotics

Vex Robotics Design Systems produces several product lines, from the Vex IQ system targeted towards elementary and middle school students to the Vex Pro product line, targeted towards adults. The product line most competitive with the robot we are developing is the Vex IQ system, so we will focus on it. The IQ line departs from their traditional metal parts system in favor of safer plastic parts. The \$250 starter kit includes 600 structural components, the “robot brain”, four motors, one gyroscope, two bumpers, one capacitive touch sensor, one distance sensor, one color sensor, and associated cables. The Vex system is much more open-ended than Mindstorms, instructions give users a few basic robots to build and add to.



*Figure 1-6: Vex IQ System*

At the heart of all Vex creations is the “Robot Brain,” which contains a 80MHz Cortex M-4 microprocessor. The Brain has 12 identical input/output ports and a radio port that allows the robot to be driven with a remote control. It is powered by a rechargeable lithium ion battery and is programmable via USB. Included rechargeable batteries give it a significant edge over the Lego Mindstorms replaceable alkaline batteries.



*Figure 1-7: Vex IQ Robot Brain*

The initial Vex kit gives students a lot of options for construction, and additional components range from the \$30 gyroscope to \$5 touch sensors (Vex Website). Additional construction parts, gears, motors, and batteries are also available. The Vex kit appears to expand into more complex robots very elegantly.



*Figure 1-8: VexIQ Gyroscope Sensor*

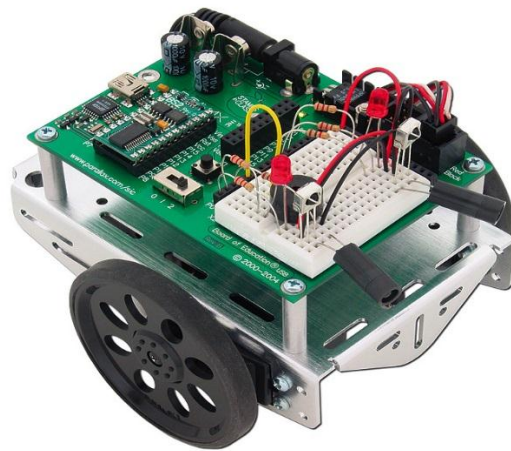
The Robot Brain comes preloaded with a few simple programs, and new ones can be constructed in either RobotC, a C-based programming language adapted by Vex, or graphically via something called ModKit. As the platform has not yet been released, more detailed information about the graphical programming interface is difficult to find. Allowing the use of a traditional, text-based programming language, though, provides a very good backup if the graphical interface does not perform well.

Vex and Lego Mindstorms share quite a few ideas. Both kits provide specialized hardware that is assembled into the physical robot and use a central microcomputer to control behavior. For both devices, the behavior of the robot is controlled by programming the microprocessor. Also in both of these devices, programming proper behavior is the single greatest weakness.



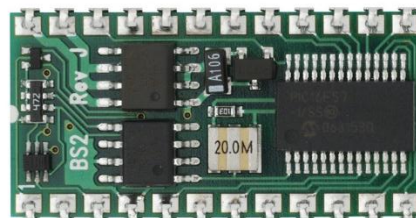
## The Boe-Bot

Parallax's Boe-Bot, or Board of Education robot, is a rather different sort of robotics platform. It is targeted towards high school and college age students, and it focuses much more heavily on electronics and programming than it does on mechanics. All of the Boe-Bots use the same basic chassis, which has two rear drive wheels and a low-friction spherical front wheel. The only real chassis modification that is available is a crawler drive system instead of a wheel-based system—overall the mechanics of the robot are not really adjustable.



*Figure 1-9: Parallax Boe-Bot*

The focus of the Boe-Bot is on modifying the electrical system and the code. The circuitry is exposed and a breadboard is mounted on the front of the device. This is used to construct various circuits, ranging from mounting the touch-sensitive whiskers to a voltage divider for a light sensor. Far more complex projects additions are also possible. The core of the stock Boe-Bot is the BASIC Stamp microcontroller, which is a 20MHz processor with 16 I/O pins. In this robot, the user directly interacts with the microcontroller.



*Figure 1-10: BASIC Stamp Microcontroller*

The robot is programmed in PBASIC, a variant of the BASIC language modified for use with the BASIC Stamp microcontrollers. The \$160 Boe-Bot beginner's kit (Amazon, Boe-Bot) comes with a booklet of sample projects that combine programming and simple circuitry. These projects range from simple motor control to navigation via photoresistor, touch sensor, and infrared distance sensor.

The breadboard prototyping interface is confusing for younger and inexperienced users—it's easy to follow the guide-book and copy the circuit onto the breadboard without actually understanding it. While the Boe-Bot has a large, active community of users on the internet, determining what to do after completing the guidebook can also be challenging.

The Boe-Bot is a very nice prototyping and development platform that is spectacularly expandable. Because of this, it is very definitely targeted towards people that already have some understanding of circuitry and programming.

## Electronics Kits

The electronics kits I have seen, played with, and researched, whether they be stationary boards that make motorcycle noises or robots with breadboards attached, do not lend themselves to particularly intuitive understanding. The real issue is that it is easy for a young person to copy the layout of wires and circuit elements from pictures, but doing that doesn't necessarily impart any real meaning. So it's quite possible to build a circuit and not remotely understand how it works. Also, electrical kits like these are quite fragile. Individual components are not current-limited, and if a student tries to experiment with a component they don't fully understand, it's highly likely that they will burn up something that can't easily be replaced.

## 130 in One Kits

There are a high number of kits of this variety. They usually cost around \$50 and include sections of resistors and capacitors, a few switches, an LCD display, a speaker, and batteries. They also come with instruction books that include hundreds of various circuits, ranging from water level detectors to ones that make motorcycle noises when you squeeze loose wires. Wires



are connected to components by inserting them into springs. This usually leads to very messy prototyping and difficult circuit debugging.



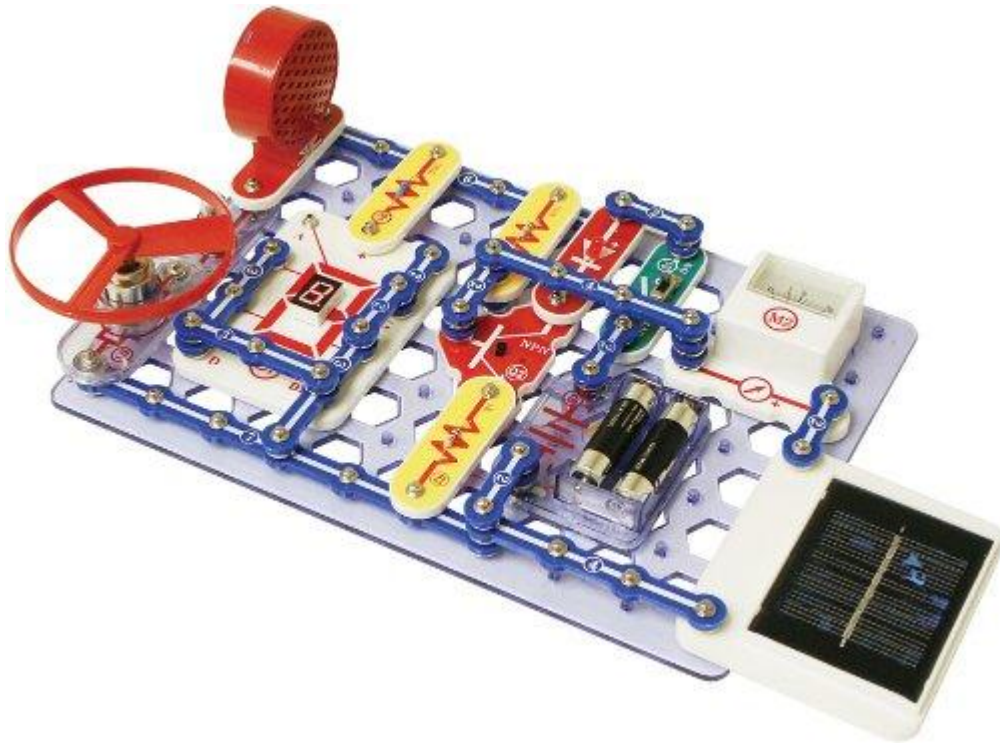
Figure 1-11:130-in-1 Electronics Kit

User reviews indicate that while these kits can be quite entertaining, it's very difficult to make the jump between copying pictures of wire layouts and following the circuit diagrams. Also, it's very easy to burn out components, ruining the entire board.

### Snap circuits

Snap circuits are a modern improvement to the spring connectors of the 130-in-1 kits.

Components are snapped onto a grid in order to make circuits. Various kits range from \$30 to \$250, and they come with a wide variety of components. Circuit assembly and component replacement is much easier, and moving around components instead of wires gives users a bit of a better sense of what they're actually using. However, components are still not protected from damage.



*Figure 1-12: Snap Circuits*

User reviews indicate that the arrangement of components can turn circuit assembly into a geometric puzzle. This removes some of the focus from the electrical aspect of the circuit, but is still fun for kids to play with. While this is certainly a fun toy, it is perhaps not the best way to actually teach children how to build circuits.

## Chapter 2 - Theory

In this section, I will give an in-depth explanation of the robot's design. First, we will examine the core components that make up the robot and explain why each was chosen. Then we will look at each building block that makes up the robot, simulate it in PSPICE, and explore the expected behaviors. Finally, we will examine the overall system design of the robot and explore the positional feedback loop, derive equations to model the robot's movement in space, perform some analysis of the performance characteristics of the robot, and finally simulate the behavior of the full control loop in PSpice.

### Core component choices

In a system designed for education, every component needs to be carefully chosen based on different requirements. While the design goals for the project demand that we restrict the components we use to resistors, capacitors, and operational amplifiers, there is obviously a need for a select few specialized components. Building an analog distance sensor out of these components would probably not work very well, there is no way at all to arrange them into an efficient motor. Thus, we will carefully pick out the more specialized components, such as the distance sensor, motor, and batteries.

### Distance Sensor

The sensor is the heart of the device. How well it works—that is to say, how quickly and reliably it responds to changes in distance—is absolutely crucial to the proper behavior of our robot. It must output some analog voltage level, be powered by some readily available voltage, and function well under different lighting conditions.

There are two main categories of distance sensors: ultrasonic and infrared. Ultrasonic sensors reflect sound waves off of a surface and pick them up with a microphone, the time delay is the distance. These sensors can detect objects within a wide range of distance, but they have a very low resolution and sometimes behave strangely while trying to resolve objects smaller than their wavelength. Another issue these suffer from is that identical units tend to interfere with each other. While some sensors intended for hobby robots, such as Parallax's Ping))) Figure

2-1), are relatively inexpensive, devices with analog voltage output seem to be restricted to higher end commercial applications and have prohibitively high prices.



*Figure 2-1: Parallax Ping)))*

Commercial uses for infrared sensors include vending machines, CD players, arcade equipment, and robotic vacuums. Thus, there are far more low-price options available for infrared sensors. A quick search on Digikey reveals sensors that detect distances around 1.5mm to a meter available for prices ranging from \$1 to \$100. Many sensors also offer analog outputs rather than serial communications, making them viable candidates for our robot.



*Figure 2-2: Sharp GP2Y0A21YK0F*

Since we expect our robot to travel at an adjustable distance between a few inches to a few feet from a wall, we chose the Sharp GP2Y0A21YK0F distance sensor, Figure 2-2. According to the datasheet, it senses distance between 4" and 31" and produces an output voltage level between 3.3V and 0V, as can be seen in Figure 2-3. Its maximum operating voltage range is 4.5V to 5.5V, and its current consumption is 30mA. This makes it a perfect fit for our robot.

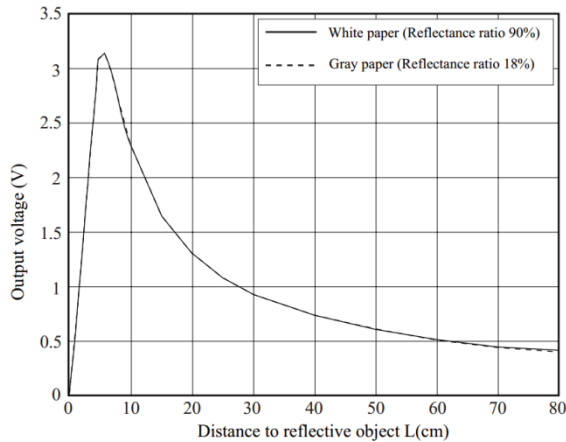


Figure 2-3: Operating Spec for Distance Sensor

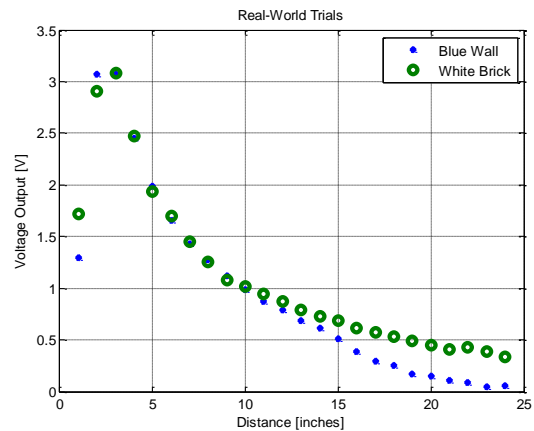


Figure 2-4: Real World Test of Distance Sensor

Since it's never safe to assume a datasheet is correct about critical performance parameters, I ran a tested the device myself. The circuit itself is quite simple—the device is connected to 5V and ground, and the output from the third pin is measured with a digital multimeter. I ran trials against a smooth blue wall in shadowy conditions, then against a rough-finished concrete block wall in full sunlight. The results can be seen above in Figure 2-4.

The device exceeded my expectations by far, the measured curves were nearly identical and are very close to the predictions in the datasheet. While absolute offset changes slightly, distance between 5 and 10 inches—where we mostly expect the robot to be—is extraordinarily consistent. Even in full sunlight, the sensor will reliably be able to detect the position of the robot. There are no artifacts or unexplained spikes, the sampling appears to be clean and nearly perfectly predictable, as can be seen in Figure 2-5: Distance Sensor Output below

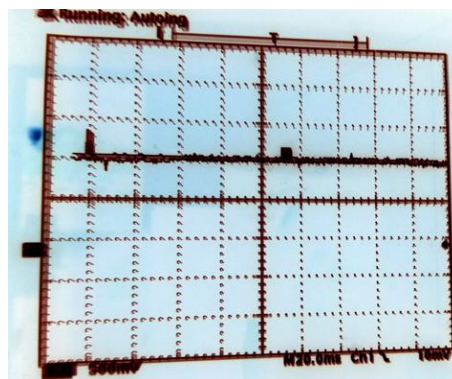


Figure 2-5: Distance Sensor Output

One issue with this sensor is that because it uses an infrared LED, careful power regulation is necessary to avoid burning it out. While all the other components in my system are fairly rugged, this one is fragile and warrants careful protection.

## Motor

The motor is another specialized component. The driving constraints for which motor we use are speed, torque, and cost. Small DC motors tend to spin at very high speeds--around 10,000 RPM. The speed of the robot at a given motor RPM can be calculated using the expansion 2-1 below, where  $d$  is the diameter of the wheel.

$$d * \pi * rpm = speed \left[ \frac{in}{min} \right] * \frac{1}{12[in]*60[sec]} = speed \left[ \frac{ft}{sec} \right] \quad (2-1)$$

So, a robot with a motor speed of 100 rpm and a 3" diameter wheel would travel 1.3 feet/second, or 0.98 miles/hour. This is slower than a person walks, but fast enough not to be boring. Let's set this as the target for motor speed.



Figure 2-6: The Motor

The motor I ended up using is 7000 RPM motor with a gear reduction down to 140RPM. The motor has a free-running current of 0.12A and a stall current of 1A. While the supplier did not provide a full datasheet for the motor, it is possible to use lab measurements and observations and develop an accurate model.

## Battery

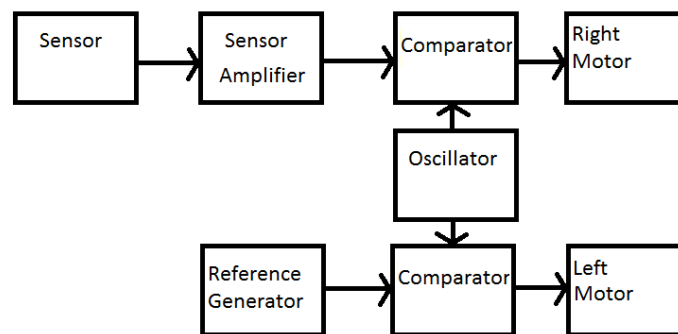
Due to properties of the logic and feedback loops, the robot uses a double-ended power supply design, +/- 6V. This makes it rather difficult to power using normal batteries, as they typically only have two terminals. For the purposes of the initial prototype, the simplest and most cost-



effective method of achieving this is to buy a composite 12V battery, open it up, and use the mid-point of the battery back as ground. In later iterations of the device, a more elegant solution may be in order, but for a prototype this will work quite nicely.

## Overview of each circuit block

Next, we will examine what makes up the robot. Even though this is a fairly simple robot, breaking it down into different functional blocks gives us a way to develop and test different parts of the system individually.



*Figure 2-7: Block Diagram*

In order to make the robot move, we will use PWM. The left wheel of the robot runs at fixed a PWM duty cycle, around 50%. The duty cycle of the right wheel, however, is controlled by the distance sensor (which also mounted on the right). When the robot gets too close to the wall, the speed of the right wheel increases in order to turn the robot away from the wall. When the robot gets too far, the speed of the right wheel decreases in order to turn the robot towards the wall.

One of the simplest ways to generate PWM is to compare a triangle wave to a DC voltage. When the triangle wave is above the DC voltage, the output of the PWM wave is high, when it is below the DC voltage, the output is low. As long as the triangle wave is fast enough, the voltage from the distance sensor can be approximated as DC and a simple comparator is perfect for generating PWM.

## Triangle Wave Generator

There are lots of different approaches to generating a triangle wave. The circuit in Figure 2-8 combines an RC oscillator and a positive feedback loop to generate both a square wave and a triangle wave.

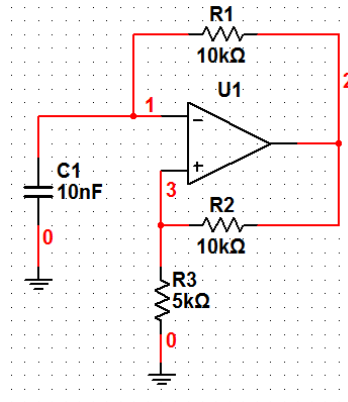


Figure 2-8: Triangle Wave Generator

While the output of the operational amplifier, node 2, is a square wave, node 1 is the exponential charge and discharge of the capacitor. If we adjust the voltage divider threshold using  $V_o = V_{in} \frac{R_2}{R_1 + R_2}$ , we can limit the capacitor charge and discharge cycle to a voltage range that is approximately linear and produce a wave that is approximately triangular. By adjusting  $R_1$  and  $C_1$ , we can adjust the frequency of the oscillator.

$$f = \frac{1}{2 \ln(3)RC} \quad (2-2)$$

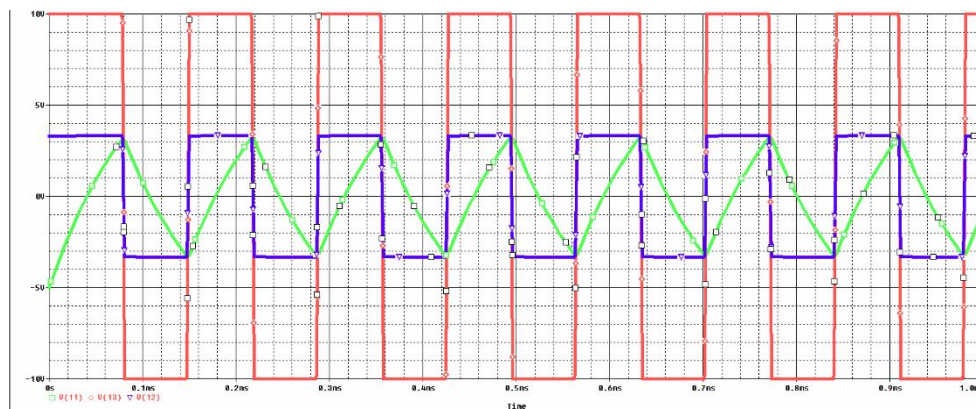


Figure 2-9: Oscillator Simulation



As we can see in Figure 2-9 (a PSpice simulation of the schematic in Figure 2-8: Triangle Wave Generator), node 1 of the oscillator produces an approximate triangle wave of 4V at 7142 kHz. Node 2, the output of the operational amplifier, produces a rail-to-rail square wave at the same frequency, while node 3 outputs a 4V peak square wave. The peak voltage of the triangle wave output can be adjusted by modifying the voltage divider, and the frequency can be adjusted by modifying the RC oscillator. The upper limit on frequency is the slew rate of the amplifier.

### Sensor Block

Now that we have our triangle wave, the next step is to make sure the voltage from the distance sensor is actually in the right range. The sensor produces voltages between 0 and 3 volts. In order to generate a 50% PWM wave though, the output from the sensor should be 0V—so whatever voltage we offset the sensor input by will be the fixed distance from the wall. To achieve this, we use a summing amplifier as can be seen in Figure 2-10.

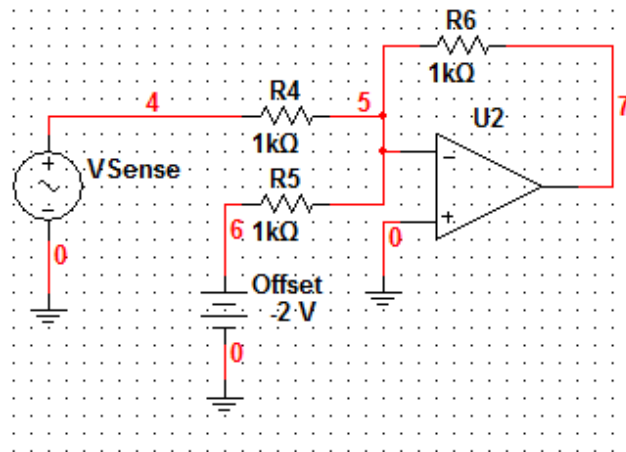


Figure 2-10: Summing Amplifier

A -2V offset put the sensor right in the middle of its distance range—any disturbance should be detected and corrected properly. A simulation of this circuit can be seen in Figure 2-11.

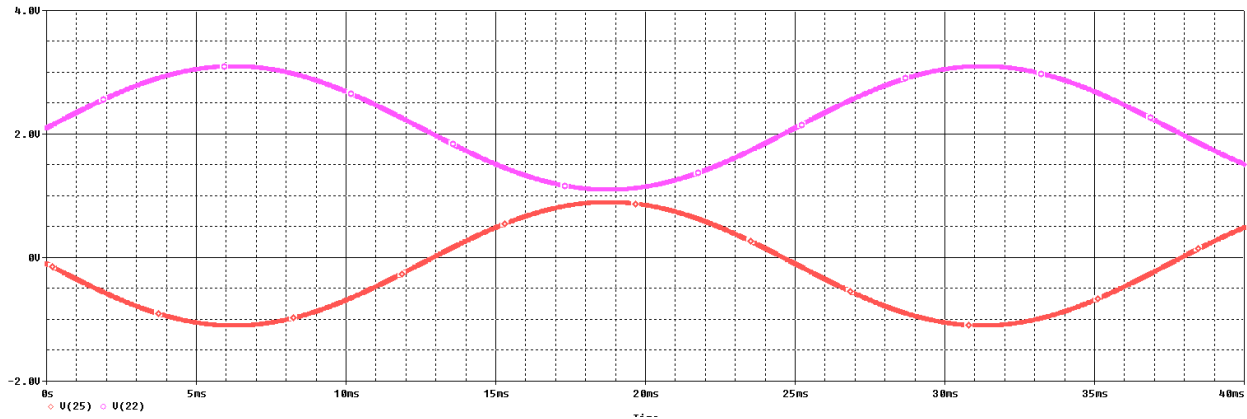


Figure 2-11: PSpice Sensor Simulation

In Figure 2-11, the top wave is the simulated sensor and the bottom wave is the output of the summer and offset circuit. For the purpose of this simulation, the sensor voltage was approximated with an offset sine wave. This is what we would see if the robot was underdamped and travelled along a sinusoidal path. Since we are testing the summer this test setup is a good model. As we can see the summer removes the offset perfectly. This particular summer is inverting. As long as our overall system contains an even number of inversions, however, this is not a problem.

## Obstacle Avoidance

While the robot can follow corners that turn inwards perfectly, corners that turn outwards and obstacles in front of the robot in general pose a different problem. We can adapt the wall follower circuit quite easily.

When following a straight wall, the left wheel and right wheel are at equal speeds. If an object appears in front of the robot, it should veer away from the wall and around the object. This is achievable simply by increasing the speed of the right wheel--the same thing that happens if we get too close to the wall. While distance in front of the robot is infinite (near 0V), the robot should follow walls normally. As it approaches an obstacle, the sensor voltage will increase. If we simply add the wall follower and obstacle avoidance voltages together, the robot should both avoid obstacles and follow walls.

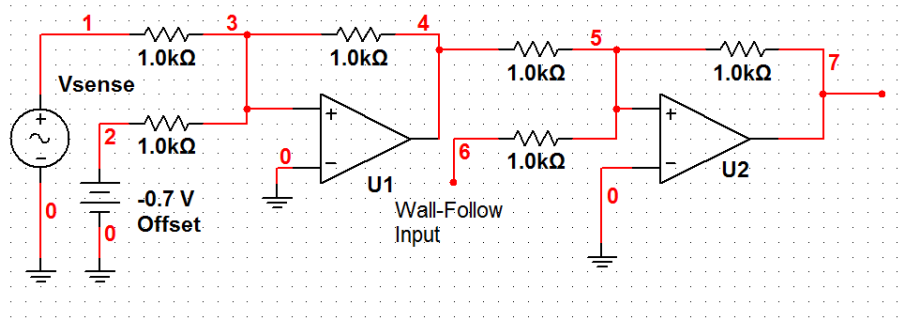


Figure 2-12: Obstacle Avoidance

This schematic (Figure 2-12) biases the sensor and sums it with the other sensor block. This is simulated in the Figure 2-13, where the obstacle is approximated by a ramp and the distance to the wall is simulated by a sine wave. The top graph is the PWM of the right wheel.

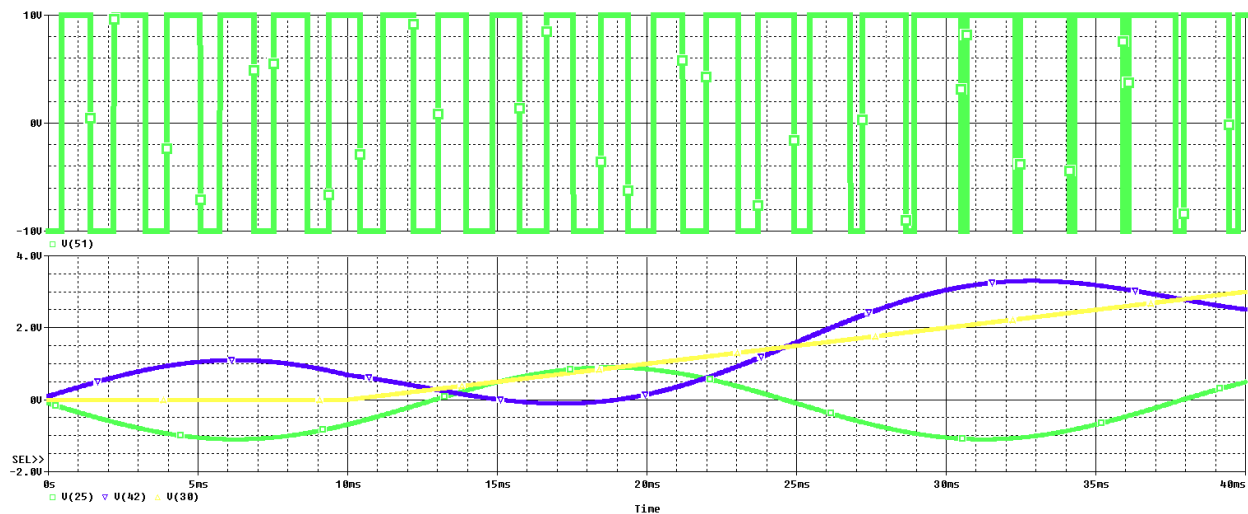


Figure 2-13: Obstacle Avoidance and Wall Following

As we can see, this produces the desired result nicely--as the robot approaches an obstacle, the duty cycle of the right wheel increases dramatically in order to turn the robot away from the angle.

### Right Wheel PWM block

The output from the summer produces a DC voltage level that's higher when we want the wheel to go faster and lower when we want the wheel to go slower. If we compare this voltage to the triangle wave generated in the first section, we will achieve a PWM wave with a duty cycle directly proportional to the input voltage level, as seen in the upper half of figure 2-13.

The maximum speed of the motor is reached at 100% duty cycle, or when the DC voltage into the comparator is greater than or equal to the maximum peak of the triangle wave. With the addition of the obstacle avoid, the maximum combined voltage we can achieve is a little over six volts--enough to clip at the op-amp rail and certainly greater than the 4V peak of the triangle wave.

### Left Wheel PWM block

Generating PWM for the left wheel follows the same principal, except our DC voltage input is set by an adjustable voltage divider instead of the outputs of two sensors. While ideally we could leave the speed of this wheel at 50% duty cycle and have everything work nicely, the additional control of the potentiometer will allow for the absolute speed of the robot to be adjusted.

### Controls Loop

This is, of course, a controls system. The robot uses negative feedback to adjust the distance to the wall. If the distance to the wall becomes too great, the right wheel slows down and the robot turns towards the wall. Similarly, if the distance becomes too little, the right wheel speeds up and the robot turns away from the wall. The controls diagram can be seen below in Figure 2-14.

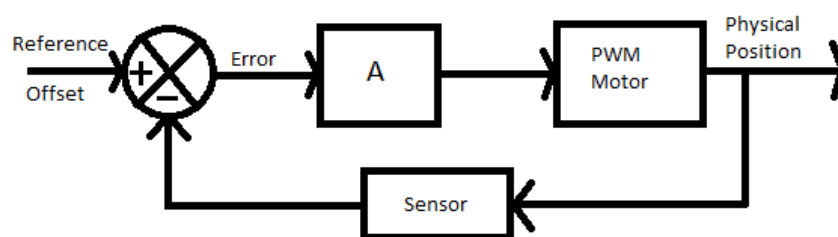


Figure 2-14: Control Loop

We need to make sure that the dampening is actually where we want it to be—the response of the robot should settle quickly with minimal oscillation.

## Deriving the position equations

In order to model this, we need to be able to plot the position of the robot in real space in PSpice. The robot uses tank steering, that is it varies the relative speed of the two drive wheels in order to turn. The front wheel freely slides. In this model, we will assume that the front wheel is frictionless. If we let  $V_R$  and  $V_L$  represent the speeds of the two rear wheels,  $w$  width of the wheelbase, and  $p$  the length of the wheelbase, we can begin to derive equations for the position of the robot instantaneously. First, let's assume that one wheel of the robot is at a fixed position and the other is turning. This clearly causes the robot to turn in a circle, with the origin at zero.

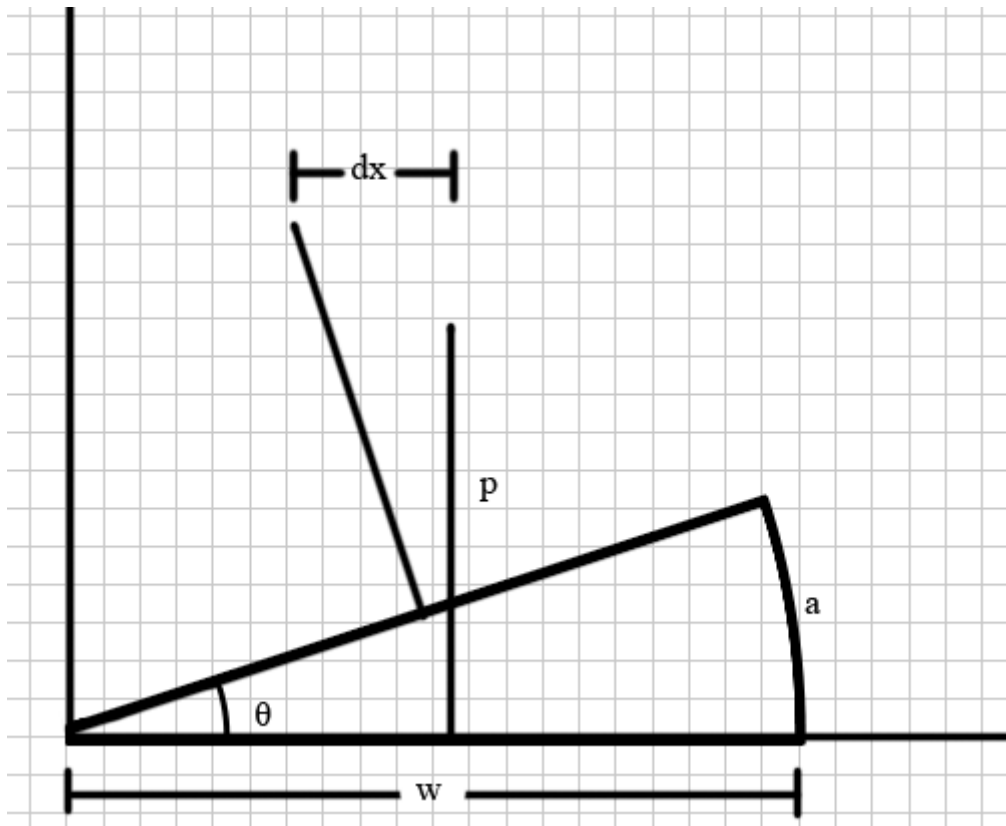


Figure 2-15: Robot at Origin

The robot travels arc distance  $a$  given the difference of the wheel speeds per unit time. By taking the difference of the two speeds, we can calculate the change in  $x$  position while ignoring motion on the  $y$  axis. This convenient trick will save us calculation time.

$$a = \frac{(V_R - V_L)dt}{w} \quad (2-3)$$

Now that we know the arc length per unit time, we can calculate  $\theta$ , the angle that the robot is pointing after traversing distance  $a$ , using equation [2-4].

$$\theta = \frac{180a}{\pi b} \quad (2-4)$$

Given this information, we can solve for  $dx = p \theta$ , the change in  $x$  distance as the robot turns, using formula 2-5.

$$dx = \frac{p(V_R - V_L)dt}{w} \quad (2-5)$$

Integrating this equation over time allows us to calculate the position of the robot in the  $x$ -axis as the robot moves. Integrating this by hand doesn't serve much of a purpose, as we need the equation in this form to simulate it in PSpice.

Finding the  $y$ -position of the robot from here is relatively simple. We know the angle the robot is travelling, the cosine of the angle gives us the component of the velocity in the  $y$  direction. The overall velocity is just the average of the two wheel velocities, as we can see in equation 2-6

$$V = \frac{V_R + V_L}{2} \quad (2-6)$$

Thus, the position of the robot in the  $y$  axis over time can be easily derived, as seen in equation 2-7.

$$dy = V \cos(\theta) dt \quad (2-7)$$

Integrating this over time gives us the absolute position of the robot in the  $y$  axis. Together, these two equations allow us to model the behavior of the robot in PSpice.

### PSPICE Control Simulations

If we plug these equations into PSpice and model the control loop, we can create a virtual wall and watch how the robot follows it. This allows us to predict how motor speeds and proportional feedback values affect the actual position of the robot. This is key to the original idea—by playing with circuit components, someone can watch how the behavior of the robot changes.

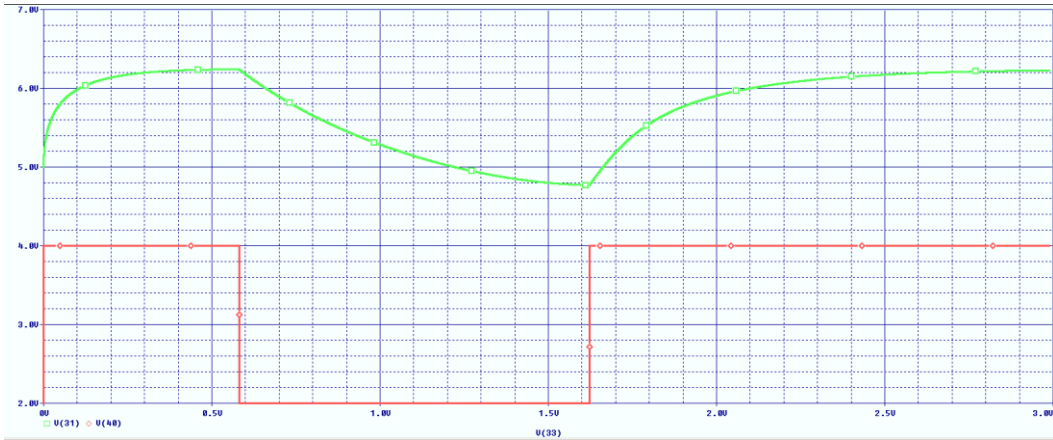


Figure 2-16: Position Tracking

In this figure, the bottom line is the wall that the robot is trying to follow and the green line is the actual path the robot will take. This simulation is over a two second interval—even though the robot’s responses look a bit on the slow side, it’s settling to follow the new path in around half a second. There is no overshoot and no oscillation after the robot settles, so this shows excellent performance.

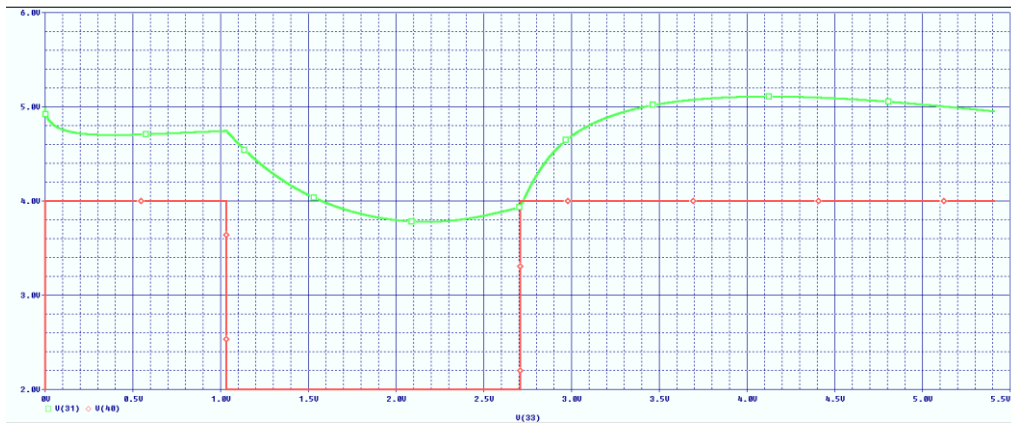


Figure 2-17: Increased Feedback

If we increase the proportional feedback, we can see that the robot responds more quickly, but that there is significant overshoot—to the point that it actually collides with the wall at one point. The ability to adjust and experiment with this, though is fundamental to the idea of the robotics platform.

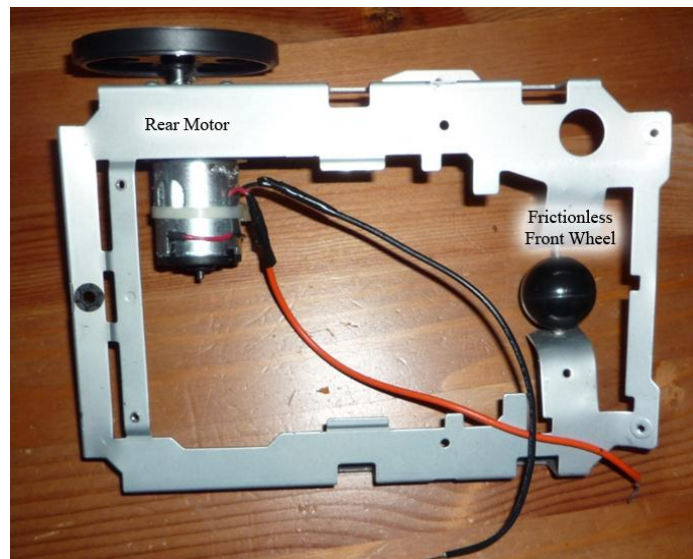
The PSPICE code for this simulation can be found in Appendix A.

## Chapter 3 - Alpha Implementation

The first prototype for the robot was simply a functionality test for the circuitry and motors. It neglects to include device protection, magnetically interconnecting components, specialized component packaging, or aesthetically pleasing design. Due to some constraints, the power source was a set of AA alkaline batteries held together by electrical tape. While this is clearly not the most elegant solution, it is good enough to verify the functionality of the device.

### Mechanical Design

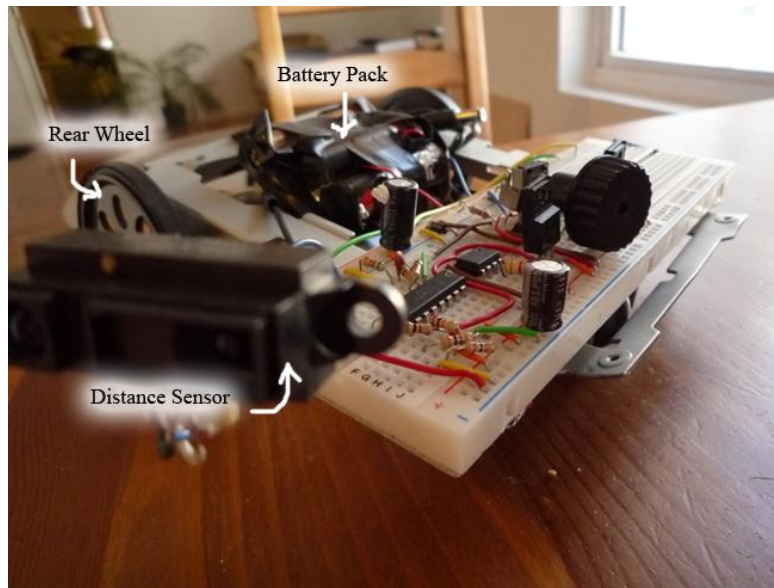
Since speed and simplicity were priority, the chassis was made from the skeleton of a modified CD-ROM drive, as can be seen in figure 3-1.



*Figure 3-1: Alpha Chassis*

The motors were screwed to the frame while the battery pack and breadboard were only attached with electrical tape. The front wheel was attached with a cotter pin and is a smooth ball intended to easily slide in any direction. Later on, this might be replaced with an omni-wheel that allows the robot to turn more easily. The rear drive wheels did not match the keying of the motor shafts, so they were temporarily affixed with hot glue.



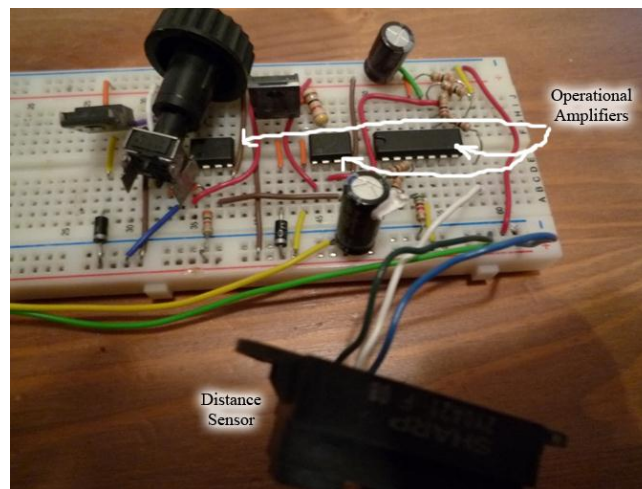


*Figure 3-2: The First Prototype*

While far from elegant, this design stayed in one piece long enough to demonstrate that the conceptual design worked very well.

## Electrical implementation

The circuitry was assembled on a bread-board, this allowed for easy prototyping and a compact, light-weight design. Eventually components will be individually packaged with signal LEDs and current protection included, prototyping the circuit on a breadboard and testing it with this system allows me to verify the real-world performance of the overall electrical design.



*Figure 3-3: Circuit Board*

## Oscillator

This implementation of the oscillator used 1000 pF capacitor and a 10K resistor to produce a 20 kHz triangle wave from -4 to +4 V, as can be seen in the oscilloscope photo Figure 3-4 below.

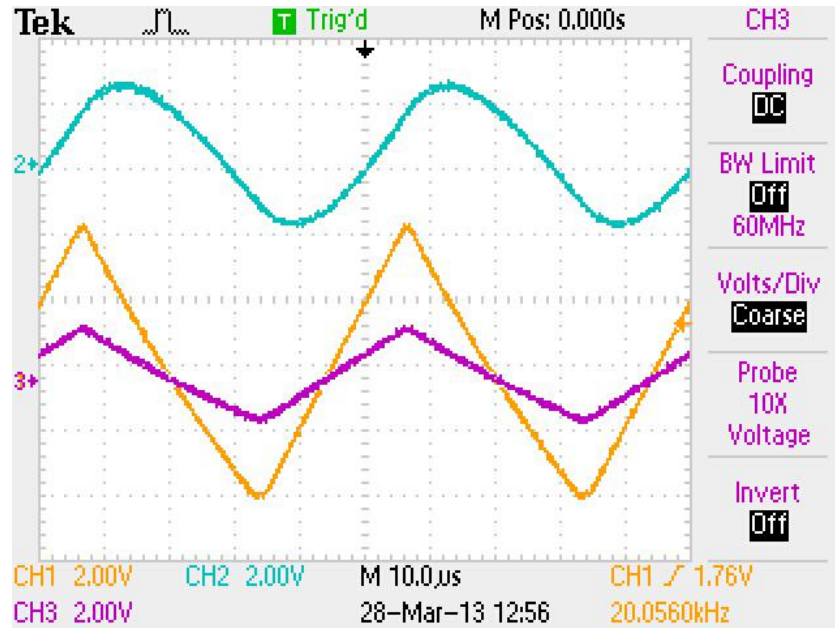


Figure 3-4: Oscillator Output

Channel 2 is where we expected to see a triangle wave output, while channel 1 should have been a square wave. Clearly something is wrong though--because this is a first-generation prototype, I've utilized the most readily available parts. This operational amplifier is actually the LM348, which is slew-rate limited to 20kHz. Our RC oscillator is tuned to 45kHz, quite a bit faster than the LM348 can handle, thus the operational amplifier is slew-rate limiting. This distorts channel 1 from the square wave we expected into a nice, linear triangle wave--exactly what we were trying to generate in the first place. Since channel 2 is distorted badly, we will simply use the channel 1 output as the main triangle wave generator for PWM generation.

## Wall Follower Offset

The wall follower adjustment circuit was implemented with 1k resistors and produces 0V at a distance of 5 inches from the wall. The voltage from the compensator circuit is nice and stable as can be seen in Figure 3-5.

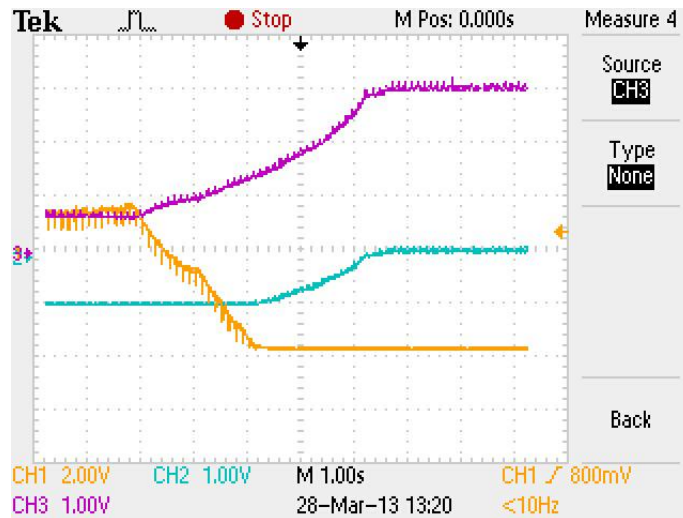


Figure 3-5: Distance Sensor as Distance Changes

Here channel 1 is the output of the summer, channel 2 is the offset voltage, and channel 3 is the sensor voltage and the sensor moves towards the wall. The offset is set to -1V, this is appropriately added to the output of the summation. We can see how as distance increases, the output decreases and the offset stays constant until the output hits the rail. This behavior is exactly as we expected.

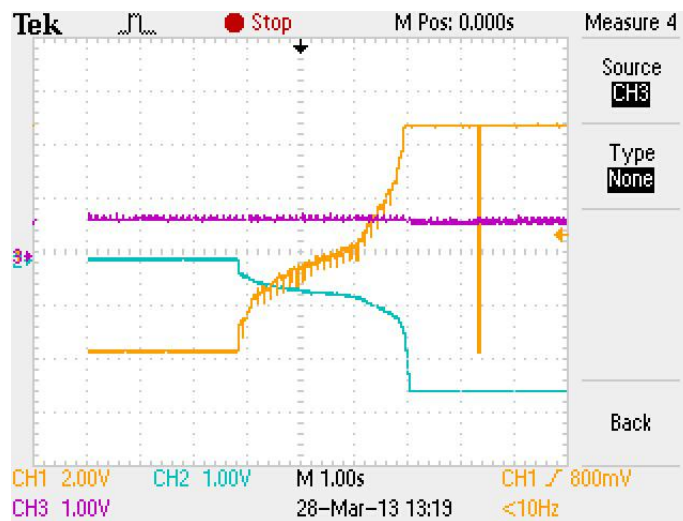


Figure 3-6: Distance sensor as bias changes

In Figure 3-6 we can see how a change in the offset voltage produces a change in the output. The sensor stays the same distance from the wall, but the output of the circuit changes as the offset changes. This can be used to adjust the robot's overall distance from the wall.

## Obstacle Avoidance

The obstacle avoidance circuit is very similar to the wall follower circuit, with the addition of the summer. In order to keep the prototype as simple as possible, this sub-circuit was omitted. According to our simulations, as long as the wall-follower circuit works well, this should too.

## PWM Generator

The comparators for generating PWM for the right wheel are simply op-amps with inputs-- there are no component values to select. The right wheel inputs are the output of the wall follower circuit and the triangle wave. The left wheel inputs are a DC voltage selected by an adjustable voltage divider and the triangle wave. The outputs from the op-amps are then sent to MOSFETS that drive the motors. The schematic for this may be seen below, with the addition of protection diodes to eliminate voltage spikes from the motors

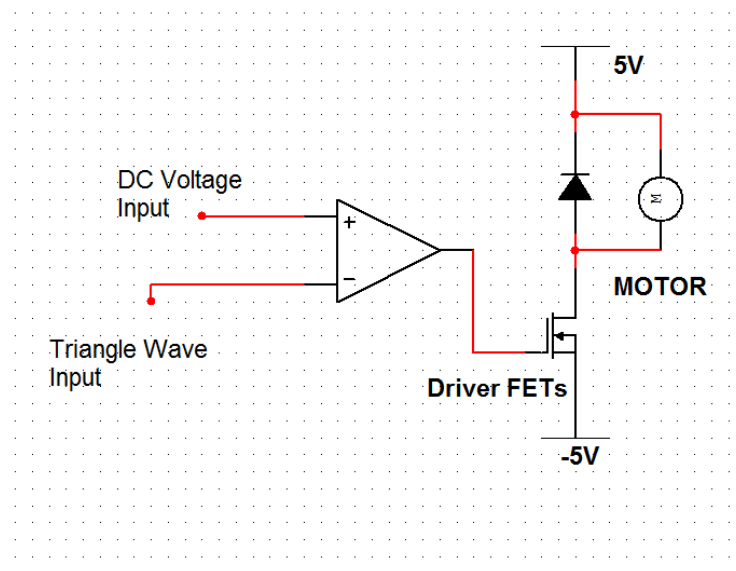


Figure 3-7: Motor Driver

This driver setup inverts the output of the PWM. In this implementation it works fine however, given the total number of inverting stages in the system.

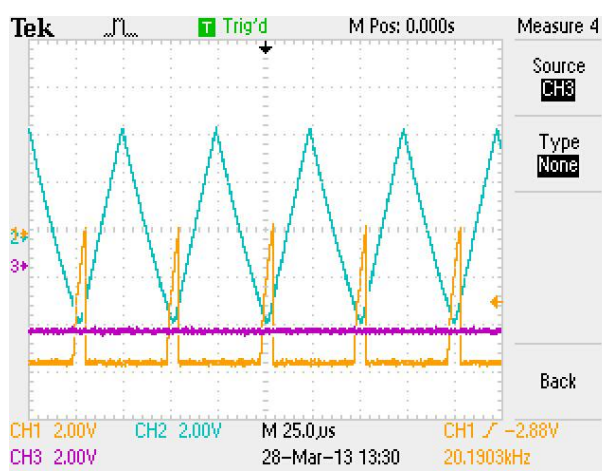


Figure 3-8: PWM at Minimum

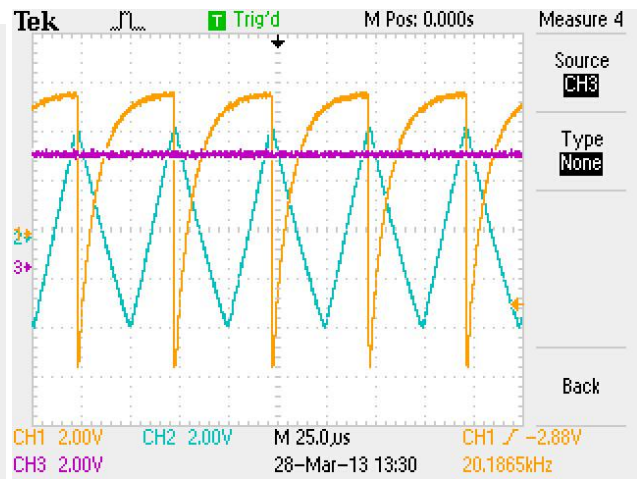


Figure 3-9: PWM at Maximum

As we can see here, the PWM output is not quite as clean as we would like it—the exponential characteristic of the MOSFET gate capacitor is very clearly present. This reduces PWM efficiency and increases heat dissipation in the MOSFETs. Due to the size of MOSFETs, however, this is not currently a problem. None the less, in the next design iteration we could fix the issue by adding a small transistor to drive the larger MOSFET.

## Results

The alpha implementation of the robot looked rather ridiculous, as is clear from the photos in the beginning of this chapter, but extensive testing revealed that it followed walls beautifully. It can even turn corners and recover properly.

Oscillation is minimal and cornering radius is reasonable. The robot tends to travel quite slowly as the motors are almost never at full speed. This can be corrected by adjusting the PWM of the left motor, however—the right motor will speed up to keep the robot on course. The only real problem with this version was that the battery tended to wiggle loose and cause unexpected behavior.

## Chapter 4 - Results and Conclusion

This project started off as an attempt to build an educational electronics platform, and ended as an in-depth look at wall-following robots. An educational platform like the one we have posed requires circuits that are both simple enough to quickly assemble and easily understand, yet complex enough for students to spend hours analyzing. The wall following robot, with its positional feedback system, fits this bill perfectly. It uses only a few operational amplifiers, two sensors, and two motors, and it can successfully navigate around a house. By changing the gain of the feedback loop, different responses will occur. The frequency of the triangle wave oscillator can be changed too, which changes how smoothly the robot drives. All of the weighting of the summer amplifiers can also be changed, allowing more sensitive response to changes in the environment and allowing for the adjustment sensitivity to objects in front of the robot than to the wall next to it.

This circuit is both simple and complex. It is simple to build and experiment with, yet it requires a significant amount of analysis and experimentation to properly understand.

This is not the end of this project. I will continue to develop the educational platform in my spare time. The wall following robot could be improved by the addition of derivative feedback, and component packaging and protection has yet to be developed. Also the alpha-prototype chassis needs to be redesigned very thoroughly before such a product could be brought to market.

This MQP carefully examined the properties of a wall-following robot and includes a mathematical derivation of the behavior of the robot and a simulation of the robot's wall-following control loop in PSpice. This lays the framework for the rest of the robotics development platform to be built in the future.

## Appendix A: PSpice Code

### Wall Following Control Loop Simulation:

final attempt

```
****BLOCK #1****
****SENSOR****
*INPUT: Position
*OUTPUT: Error:          2
*VSense  1 0 SIN(.1 1 40)
ESense 1 0 VALUE={1*(V(31)-3)}
RSense 1 0 1Meg

****BLOCK 2****
****MOTOR****
*Input: Error            2
*Output: Velocity       16
Ev 10 0 VALUE={V(1)}
*Vmv 10 0 SIN(8, 5, 1)
Rm1 10 11 3
Emv 12 11 VALUE={2*I(Vmd)} * K*I(V)
Vmf 12 0 DC 0

Emt 13 0 VALUE={2*I(Vmf)} * K*I(Vmf)
L1 13 14 .003
R2 14 15 3
Vmd 15 0 DC 0

Eout 16 0 VALUE = {1.5*I(Vmd)}
Rout 16 0 1MEG

****BLOCK 3****
****Angles****
*Input: velocity         16
*Output: Displacement    21
Eangel 20 0 VALUE={ (V(16)-4) }
Rangel 20 21 1
Cangel 21 0 .5

****BLOCK 4****
****Position****
*Input: displacement 21, wall
*Output: position    31
Expos 30 0 VALUE={V(40)-V(21)}
Rxpos 30 31 .5
Cxpos 31 0 1 IC=5
Eypos 32 0 VALUE={ (1+(V(21) + 5)/2*cos(V(21)-5)) *Time}
Rypos 32 33 1
Cypos 33 0 1 IC = 0

****BLOCK 5****
*****WALL*****
*INPUT: NONE
*output: Wall
*Ewall 40 0 VALUE={4}
VWall 40 0 PULSE=(2V 4V 1s 0s 0s ,5s 10s
Rwall 40 0 1MEG
```

```
.TRAN 10us 7s 0s 100us
.PROBE
.END
```

## Overall Circuit Simulation:

Smooth Moter Output Attempt

```
*Main Supply
VCC 0 1 DC 10
VEE 2 0 DC 10
R1 1 0 1MEG
R2 2 0 1MEG

*Triangle Wave Block (Oscillator) (output 11)
*CT 0 11 10n IC=5
*RT1 11 13 10k
*RT2 12 13 10k
*RT3 12 0 5k
*XT 12 11 13 0 OpAmpSat
VTri 0 11 PULSE(-4 4 0 .09ms .09ms 0us 0.18ms)
RTri 11 0 1MEG

*Sensor block (Wall Follower) (output 25)
VWbias 21 0 DC 0
*VWSense 22 21 SIN(0 .5 10)
EWSense 22 21 VALUE={V(900)}
VWOffset 0 23 DC 3
RW1 22 24 1k
R5 23 24 1k
R6 24 25 1k
X3 0 24 25 0 OpAmpSat

*Other Sensor Block (Output 34)
*VObstacle 30 0 PULSE(0 3 10m 30m 0m 0u 40m)
*Vbias2 30 31 DC 2
*Voff 32 0 DC 2
*RO1 31 33 1K
*RO2 32 33 1K
*RO3 33 34 1K
*XO 0 33 34 0 OpAmpSat

*Sum the sensor blocks (Output 42)
RS1 25 41 1K
RS2 0 41 1K
RS3 41 42 1K
XS 0 41 42 0 OpAmpSat

*Make it into PWM (Output 51)
XP 42 11 51 0 OpAmpSat
RP 51 0 1MEG

*Left Wheel! (output 62)
RL1 2 61 10K
RL2 61 1 10K
```



```

XL 11 61 62 0 OpAmpSat
RL3 62 0 1MEG

*Low Pass for Right Wheel (output 70)
RLP 51 70 1000
CLP 70 0 10.59u IC = 0
*VR 0 70 2V
*RR 0 70 1Meg

*Low Pass for Left Wheel (output 80)
RRP 62 80 1600
CRP 80 0 10.59u IC = 0
*VL 0 80 1V
*RL 0 80 1Meg

**** Analysis Section ****
*Backup Posn Value
*Eoldx 999 0 VALUE={V(1002)}
*Roldx 999 0 1MEG
*Eoldy 998 0 VALUE={V(1003)}
*Roldy 998 0 1MEG

*Rotational Angle of Robot--Should I add previous theta somehow?
ErotT 1000 0 VALUE={ (V(80) - V(70))/1}
RrotT 1000 0 1MEG

*Eh?
EFvel 1001 0 VALUE={ (V(70) + V(80))/2}
*RFvel 1001 0 1MEG
*Thing
*EThing 1001 0 VALUE={ ((V(70)+V(80)) / (2*(V(80)-V(70)))) }
RThing 1001 1 1MEG
*x position
*EXposn 1002 0 VALUE={V(1001)*(cos(V(1000)))}
*RXposn 1002 0 1MEG
*y position
*EYposn 1003 0 VALUE={V(1001,0)*(sin(V(1000)))}
*RYposn 1003 0 1MEG

****Debugging****
*ETime 90 0 VALUE={Time}
*RTmie 90 0 1MEG
*EX 91 0 VALUE={Time}
*RX 91 0 1MEG
*EY 92 0 VALUE={1}
*RY 92 0 2MEG

***Take 2***
Exmagic 1002 0 VALUE={V(1001)*cos(V(1000))}
Rxicmagic 1002 1003 1
Cxicmagic 1003 0 1 IC = 0

Eymagic 1004 0 VALUE={V(1001)*sin(V(1000))}
Rymagic 1004 1005 1
Cymagic 1005 0 .1 IC = 5

***Wall***

```

```

VWall 40 0 PULSE=(4V 2V 0s 0s 0s 100ms 300ms)
Rwall 40 0 1MEG

**Sense
Esensor 900 0 VALUE={V(1003)-V(40)}
Rsensor 900 0 1MEG

*Basic Opamp courtesy uta.edu pspice tutorials
.SUBCKT OpAmpSat non inv out com
Ri non inv 1MEG
Ro int out 1.0
Et int com TABLE {V(non,inv)}=(-1mV,-10V) (1mV,10V)
.ENDS

.TRAN 10ms 300ms
.PROBE
.END

```

## Image Credits:

All URLs accessed 4/25/2013

K'Nex: [http://global.knex.com/au/images/club/ideacenter/car\\_lg.jpg](http://global.knex.com/au/images/club/ideacenter/car_lg.jpg)

NXT Brick: Inside NXT Brick: [http://jstuber.net/lego/nxt-programming/nxt-top\\_big.jpg](http://jstuber.net/lego/nxt-programming/nxt-top_big.jpg)

NXT robot: [http://commons.wikimedia.org/wiki/File:Lego\\_Mindstorms\\_Nxt-FLL.jpg](http://commons.wikimedia.org/wiki/File:Lego_Mindstorms_Nxt-FLL.jpg)

NXT Brick: [http://commons.wikimedia.org/wiki/File:Lego\\_Mindstorms\\_NXT\\_2.0-Stein\\_mit\\_%C3%BCber\\_Kabel\\_angeschlossene\\_Sensoren.JPG](http://commons.wikimedia.org/wiki/File:Lego_Mindstorms_NXT_2.0-Stein_mit_%C3%BCber_Kabel_angeschlossene_Sensoren.JPG)

Amazon Mindstorms: <http://www.amazon.com/LEGO-4544091-Mindstorms-NXT-2-0/dp/B001USHRYI>

Cite Mindstorms pricing site: <http://shop.lego.com/en-US/Light-Sensor-9844>

NXT Screenshot: [http://ph-elec.com/wp-content/uploads/2011/08/Move\\_C.jpg](http://ph-elec.com/wp-content/uploads/2011/08/Move_C.jpg)

Vex Website: <http://www.vexrobotics.com/vexiq/products>

Boe-Bot Image: <http://www.parallax.com/Portals/0/Images/Prod/2/288/28832a-L.jpg>

Basic Stamp Image:

<http://www.parallax.com/Store/Microcontrollers/BASICStampModules/tabid/134/ProductID/1/List/1/Default.aspx?SortField=UnitCost,ProductName>

Boe-Bot: <http://www.amazon.com/Parallax-Boe-Bot-Robot-Kit/dp/B000GE8RQO>

Ping: <http://www.parallax.com/tabid/768/productid/92/default.aspx>

Distance Sensor: [http://www.sharpsma.com/webfm\\_send/1489](http://www.sharpsma.com/webfm_send/1489)

Motor: <http://www.mpja.com/12VDC-140-RPM-Gearhead-Motor/productinfo/18746%20MD/>