View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by The University of Akron

The University of Akron
IdeaExchange@UAkron

Honors Research Projects

The Dr. Gary B. and Pamela S. Williams Honors College

Spring 2015

Autonomous Robot Sphere

Robert Haver
The University Of Akron, rdh20@zips.uakron.edu

Melissa Haver
The University Of Akron

Daniel Madden
The University Of Akron

Noah Robertson
The University Of Akron

Please take a moment to share how this work helps you through this survey. Your feedback will be important as we plan further development of our repository.
Follow this and additional works at: http://ideaexchange.uakron.edu/honors_research_projects


Part of the Electrical and Electronics Commons, and the Robotics Commons

Recommended Citation
Haver, Robert; Haver, Melissa; Madden, Daniel; and Robertson, Noah, "Autonomous Robot Sphere" (2015). *Honors Research Projects*. 121.
http://ideaexchange.uakron.edu/honors_research_projects/121

This Honors Research Project is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

Robert Haver

Honors Research Project / Senior Design

Spring 2015

Autonomous Robot Sphere

Over the course of the Fall 2014 and Spring 2015 semesters, I have participated in the design and implementation of an autonomous robot sphere. The autonomous robot sphere consists of a omni-wheel robot contained in a 3D-printed sphere. A transmitter, worn by the user, emits radio and ultrasonic signals which is used by the robot to locate and track the user.

Although I have been assigned the role of software manager, individual contributions have been very flexible. As software design for the project has been limited to small amounts of coding for an Arduino microcontroller, the majority of my contributions to the project have related to the mechanical design of the robot. This has primarily included the design and acquisition of a 3D-printed spherical enclosure, and 3D-printed platforms for each of the individual systems for the robot. This also included designing and implementing a stabilizer system to prevent the robot from landing on it's back inside the sphere and to maintain constant pressure on the wheels. I have also assisted in the construction and troubleshooting of our motor controller boards, as well as minor contributions on other systems.

The primary issue we encountered with this project was the motors. As the project evolved and became more complex than initially anticipated, the robot turned out to be significantly heavier and larger than expected. As a result, our motors were not powerful enough to move the robot from rest. Larger motors had to be ordered, and a custom motor controlled had to be assembled. The robot's receivers have also had issues performing as anticipated and cannot currently locate the transmitter, but this issues seems to be resolved and all other systems are functioning properly. The following report contains more details on this project and the function of each individual system.

Senior Design Team 05
Autonomous Robot Sphere
Final Report

Melissa Haver

Robert Haver

Daniel Madden

Noah Robertson

Igor Tsukerman

April 17nd, 2015

Table of Contents

Table of Figures

Table of Tables

1. ABSTRACT

The Autonomous Robot Sphere is an interactive robot toy meant to entertain kids. The robot will locate its target and execute algorithms to autonomously evade or chase a child. The sphere will contain a platform equipped with four omni-wheels, which will allow the sphere to maneuver and change direction almost instantaneously. The robot will be configured to maintain a fixed distance from the transmitter, allowing it to chase or evade the child in response to their movement. The primary advantage of our design lies in its capability to quickly adapt to changes in direction. (Melissa Haver)

2. PROBLEM STATEMENT

Need:

Kids today do not have enough toys. Technology is often used to create new and dynamic toys for children, but—especially for toddlers—most of the toys developed thus are largely sedentary, where the child sits and pushes a button to initiate flashing lights and noise. A better interactive toy would impel that child to run, twist, and roll—exercising gross motor skills, enhancing early cognitive development, and just burning off some of her or his boundless energy for the day. (Noah Robertson)

Objective:

Our project will fill this need by augmenting one of the most basic of children's toy: a ball. We will develop an autonomous robot contained completely within a soft sphere (i.e., with no external sensors or actuators) that can dynamically interact with a toddler. This ball must track the location of the child to chase after or evade him or her based on his or her behavior (such as running away or running toward). It must also detect obstacles in its path and attempt to navigate around them. The design must be robust enough to handle the bumps and tumbles associated with toddler play, and fast and responsive enough to provide engagement and excitement. (Noah Robertson)

Research Survey:

David Premack, an experimental psychologist famous for his "theory of mind" concept of how humans infer the metal states of others, did many interesting experiments which found that human children are hard-wired, practically from birth, to find objects that move with perceived intention more engaging than objects which stand still or move in straight lines, and even more engaging than objects which follow classical laws of motion or are moved by an obvious external agent (like an adult) [3]. It is with this theory of mind in mind that we chose a simple ball (which most toddlers will have a strong expectation to behave unintentionally) to automate and to add the intentional goals of chase and evasion.

Though there are a surprising number of academic and commercial robots contained within spheres—including some specifically designed as toys—we did not find any which were suitable for a user of our intended age range and which fulfilled our mission of impelling movement rather than being operable while sedentary.

The commercially available Sphero [1] is possibly the most complete development of a spherical robot. It can roll in any direction, measure distance

traveled over the ground, and detect collisions with obstacles. Though the motion mechanics seem very well designed and possibly useful to our purpose, Sphero is not sufficient for our goals because it is remote-controlled and lacks the ability to track the relative position of an external person.

The robot which most closely fulfills our overall mission is Roball [2], an autonomously-mobile sphere with interactive capability. Roball was specifically designed as a tool to measure young toddlers' development of Premack's theory of mind—specifically their perception of intention in an autonomous robot. The designers of Roball also chose to use a ball due to children's familiarity and natural engagement with the toy. Their ball does have some interactive capability, but it is linguistic, not dynamic: The Roball can ask the child to perform a task with it ("push me") and reward the child for compliance with verbal acknowledgement ("yippee") and flashing lights. The movement of Roball, though non-linear, is all preprogrammed. In many of their trials, children simply held the ball down to hear it talk and see the lights flash or sat and waited for it to come back to them when it ran its preprogrammed loop. Our design hopes to eliminate these sedentary options be requiring the child to throw or chase the ball, which will respond in real time to the child's movements.

There are two major systems which must be developed to make this toy a reality. The first is a drive system which can roll the ball in any given direction from inside the sphere; and the other is a method of spatially locating the child from the ball. Current research and development for realizing each of these technologies is discussed below.

Mechanical Drive Systems

Humans have been creating autonomous spherical robots for over a century [4]. The earliest models were spring-powered with a hanging counterweight to force the torque from a central shaft to the shell of the ball. These designs had only one degree of freedom. In the 1950s, electric motors replaced springs, but the basic design remained the same. In the 1970s, one began to see small wheeled vehicles placed freely inside a hollow shell to allow two degrees of freedom (U.S. Patent 3,722,134) some of which incorporated radio-control and structural supports to keep the drive vehicle from falling over (U.S. Patent 4,927,401). Another designs used the basic design of the early fixed-shaft spring models, but with the addition of a tilt-able counterweight to facilitate steering (U.S. Patent 6,227,933). [4]

Sphero's drive system is described in one of the most recent relevant patents (U.S. Patent 8,571,781). It uses two drive wheels held in continuous contact with the inner ball surface by a spring-forced mechanism which forces a roller-pad against the opposite side. The Sphero uses an active-feedback gyro-stabilization system to keep the two-wheeled robot upright inside the sphere. Position, velocity and orientation are determined by three sensors: a three-axis gyroscope, a three-axis accelerometer, and a three-axis magnetometer.

Range and Direction Finding

The most novel feature of our device will be its ability to react to the movement of a child. This presents a difficult design challenge because the rolling

outer frame of the sphere prevents any sensors to be mounted externally and opaques optical sensing methods. To that end we researched radio detection methods, assuming some sort of radio transmitter which can be held by or clipped to a child.

Radio direction finding (RDF) is a problem as old as the discovery of radio waves. Radio waves emitted from an antenna move radially away from the point of origin at constant speed with intensity decaying with the inverse-square of distance. At wavelengths greater than about 10 wavelengths, this radial wave can be approximated as a plane wave propagating directly away from the transmitter. This means that at any point in space, the angle of approach (AOA) of radio waves (i.e., the location of the transmitter relative to the receiver), can be determined by measuring the orientation of this plane wave.

The earliest and most basic technique developed to solve this problem is a single, rotatable loop antenna. A loop antenna senses the magnetic flux passing through it, therefore the greatest voltage is induced when the loop is parallel to the plane wave and no voltage is induced when it is perpendicular to the plane wave. Thus by rotating the loop to sense the zero-voltage angles (or nulls), the direction to toward the source can be determined. AOA can be determined by the same principle with two stationary loop antennas by orienting them orthogonally. The angle of the signal source in that case is simply the arctangent of the ratios of the induced voltages in each. More elaborate arrangements of additional loops are also available which can eliminate the 180° ambiguity in AOA. [8]

Using arrays of dipole antennas, called Adcock antennas, the electric field is detected and can be used to find AOA very similarly to the loop antennas described above. They tend to be larger and less sensitive than comparable loop antennas. [8]

For our specific design, we need a very small sensor which can be operated accurately on a moving and within a rotating shell. While a simple crossed-loop antenna may be enough, we also found some more novel designs that were built with similar constraints in mind.

In [5], a small sensor was developed for use on un-staffed aerial vehicles (UAVs) where size is also a concern. This system uses a small cluster of vector-sensing antennas to detect the AOA of EM waves. Custom calibration needed to make this technique accurate is also described.

In [6], Zhang and Lu demonstrate great success with a small-aperture technique inspired by the excellent sound-localization ability of a species of fly. They overcome the difficulties of mutual coupling present in the original proposal for this technique.

Another study designed and tested a small-scale triangulation technique with three antennas placed 24 cm apart [7]. They were able to localize AOA to within 50 degrees using magnitude differences only. By incorporating phase difference information, they were able to reduce AOA error to less than 0.5 degrees.

The majority of RDF design is created for much higher accuracy than this project will require. Since our only goal in knowing the direction to the child is to move away from her or him, it should be enough to detect AOA probably not more than by octants. Given the much higher resolution achieved by many groups with similar size constraints, there are many design options open to us. (Noah Robertson)

Marketing Requirements:

1. Fast enough to avoid targets
2. Smooth covering
3. Shock Proof
4. Rechargeable Batteries
5. Responsive to sudden changes in direction/acceleration
6. Able to track bearing and range to operator
7. Robust
8. Ability to compensate for drops and collisions
9. Shell covering will allow for easy access to interior components while remaining childproof.
10. Exterior should have no protuberance

(Dan Madden/Robert Haver)

Objective Tree:

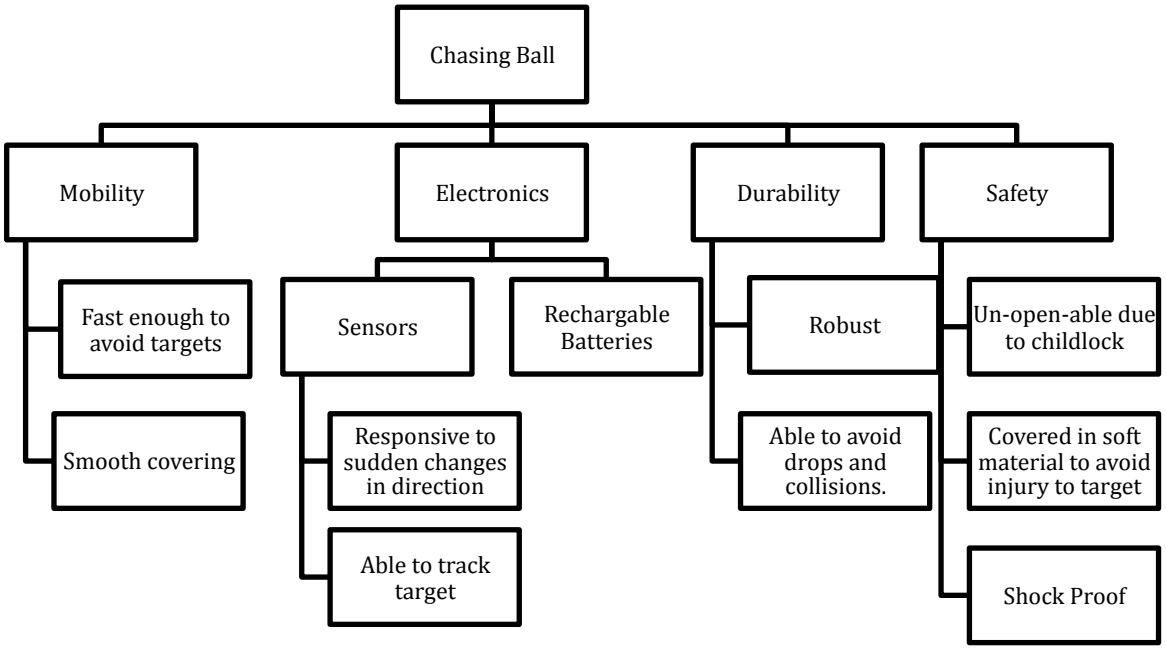


Figure 1: Objective Tree

(Dan Madden)

3. DESIGN REQUIRMENTS SPECIFICATIONS

Table 1: Design Requirements Specifications

Marketing Requirements	Engineering Requirements	Justification
1	Ball must be able to achieve and sustain a top speed of at least 5 meters per second.	To ensure that the ball is fast enough to evade a child.
1	Ball will be capable of achieving a speed of 3 meters per second within 2/3 of a second.	To ensure that the ball is capable or maintaining a 2 meter distance from target.
6	Ball must be able to resolve a bearing to the operator within ±40°.	To accurately locate direction of target.
6	Ball must be able to detect range changes of at least 100mm.	To accurately track motion of target.
6	Ball must be able to establish a bearing and range to the operator up to 10m away.	To ensure that operator can be tracked from a reasonably large distance away.
5,8	Robot will be capable of detecting when it has encountered a drop or collision.	To ensure that the ball can compensate for impacts and return to normal operation.
2,3,7,10	Robot will be fully contained in a smooth, hollow shell.	To ensure that the robot is easily able to maneuver within the casing.
5	Robot must be capable of making a 90° change in direction within 2 seconds at top speed.	To ensure that the robot can chase/evade a child in any direction.
4,5	Robot must be able to operate continuously with no external power source for at least 1 hour.	To allow for cordless, rechargeable/replaceable power.

(Robert Haver)

4. ACCEPTED TECHNICAL DESIGN

Level 0 Design:

At the highest level, this system consists of two main units (see Figure 2): The primary unit is a controller which consists of a platform having radio and ultrasound receivers from which the child's position is inferred. This unit processes child location information to drive the wheels. The secondary unit is a transmitter which is worn by the child and broadcasts the radio and ultrasonic signals to the main controller. The functional requirements of each block in Figure 2 can be found in Table 4.

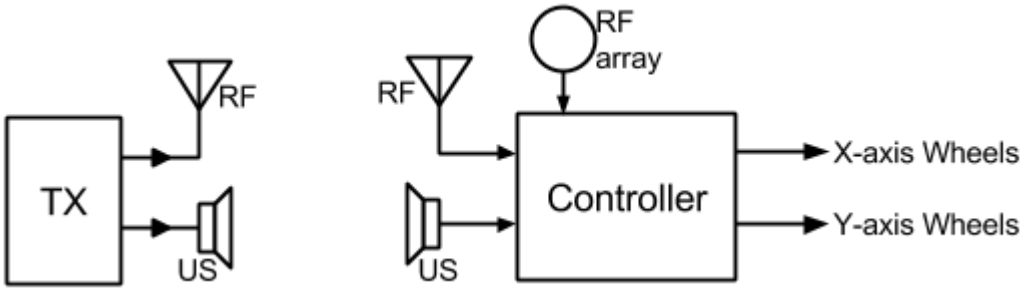


Figure 2: Level 0 Block Diagram.

(Noah Robertson)

Transmitter Level 1 Design:

To determine the range to the child, the transmitter periodically broadcasts simultaneous radio and ultrasonic pulses as shown in Figure 3.

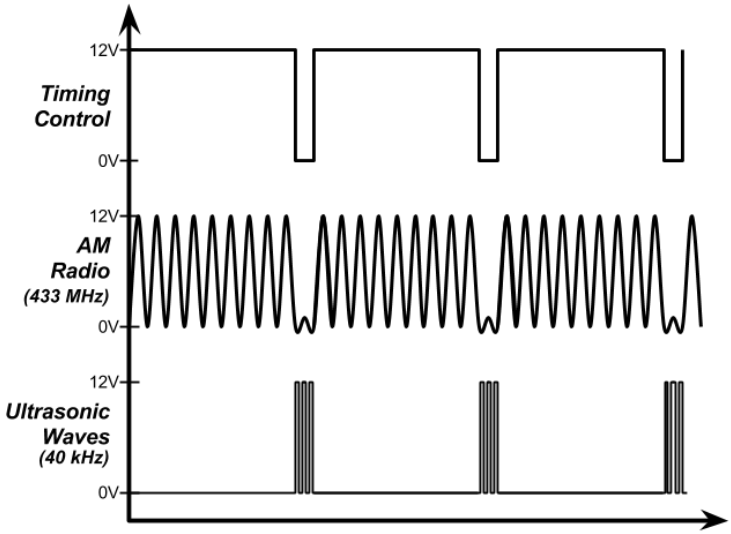


Figure 3: Transmitter Ultrasonic Pulses

The central timing controller coordinates the radio and ultrasonic pulses, generating zero-amplitude ("off") pulses on the AM carrier signal, and ultrasonic bursts ("on" pulses) on

the ultrasound transducer. By timing the difference, Δt, between when the radio "off" signal first arrives (nanoseconds after transmission) and when the sonic "on" signal arrives (milliseconds after transmission), range to the child can be determined as

$$Range = \Delta t \times vs\ , \tag{1}$$

where *vs*= 340.29 m/s is the speed of sound. Because the speed of radio is roughly six orders of magnitude higher than the speed of sound, the propagation time of the radio signal is negligible.

The level 1 diagram of the transmitter is shown in Figure 4. A central timing controller coordinates the radio and ultrasonic pulses. A short pulse (10% duty cycle) is broadcast at 10 Hz. This allows a maximum range of

$$340.29\text{m/s}\ 10Hz\ = 34.0\ meters\ ,$$

(**Error! Bookmark not defined.**)

several times farther than our expected operating range.

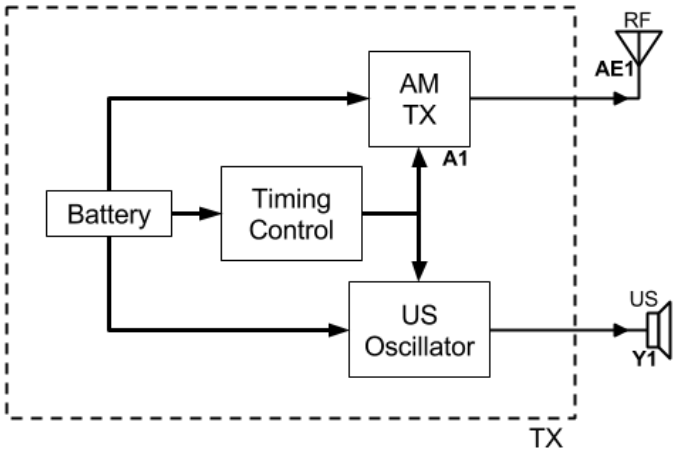


Figure 4: Transmitter Level 1 Block Diagram

A monolithic amplitude-modulation transmitter is used to drive a transmitting antenna with a 433 MHz carrier wave. The timing pulse is encoded as a binary low (zero amplitude) signal encoded in the carrier wave for 1/10th of the cycle (10ms). The rest of the cycle transmits a binary high (maximum amplitude) signal which is used for radio direction finding on the main unit.

The ultrasonic oscillator operates inverse to the radio, remaining off for the long part of the timing cycle and oscillating the ultrasonic piezo speaker at 40 kHz for the short (10ms) part. The functional requirement of each block in Figure 4 can be found in Table 5.

(Noah Robertson)

Transmitter Timing Control Level 2 Design:

To generate the 10 Hz, 90% duty cycle signal needed for the synchronized radio and ultrasonic pulses, a 555 timer is operated in astable mode as shown in Figure 5. The time of high output is

$$thigh = \ln(2) \times (787\ k\Omega + 97.6\ k\Omega) \times 150\ nF = 92.0\ ms\ , \quad (2)$$

and the low time is

$$tlow = \ln(2) \times 97.6\ k\Omega \times 150\ nF = 10.1\ ms\ , \quad (3)$$

for a realized period of 102.1 ms, a frequency of 9.79 Hz, and a duty cycle of 90.1%.

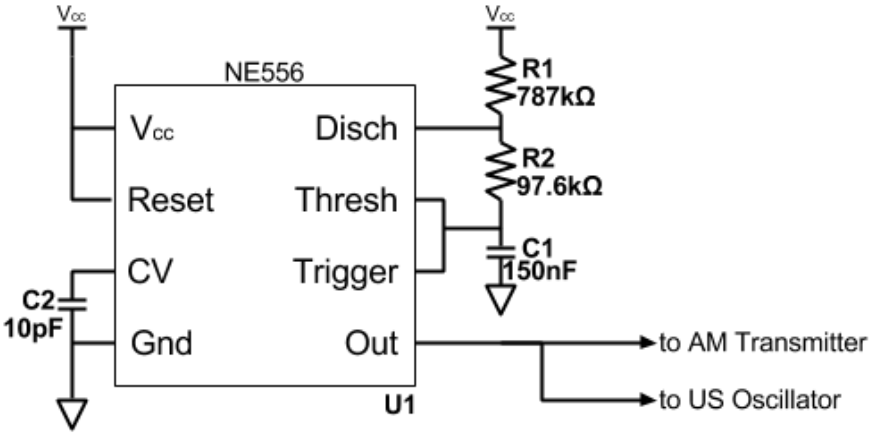


Figure 5: Transmitter Timing Control Level 2 Block Diagram

(Noah Robertson)

Transmitter US Oscillator Level 2 Design:

The ultrasonic oscillator in Figure 6 is composed of three separate, cascaded stages. Starting from the left, the first stage is an inverter which takes the 10 Hz, 90% duty cycle output of the timing control circuit and converts it into a 10 Hz, 10% duty cycle signal to the reset pin of the second stage. The second stage is a 555 in astable mode set with high time

$$thigh = \ln 2 \times 2.26\ k\Omega + 16.9\ k\Omega \times 1\ nF = 13.3\ \mu s\ , \quad (4)$$

and low time

$$tlow = \ln 2 \times 16.9\ k\Omega \times 1\ nF = 11.7\ \mu s\ , \quad (5)$$

which yields a frequency of 40.0 kHz. Since the 555 resets low, this frequency is only seen during the 10 ms that the reset pin is high, resulting in a 10 ms pulse of sound once every 100 ms. The last stage is a switching transistor which drives the piezo speaker. Since the piezo speaker has a natural resonant frequency, the 16.9kΩ resistor is replaced

with a trimmer pot to tune the frequency to maximize the sonic power from the transducer.

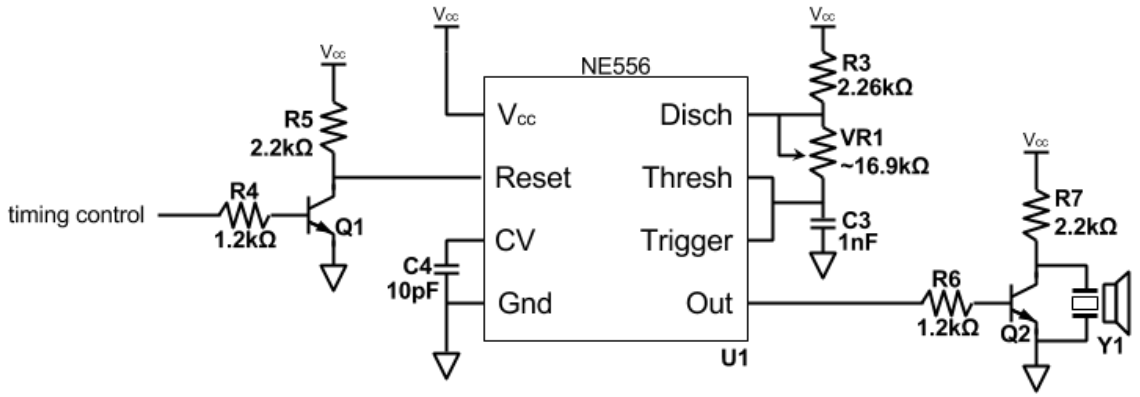


Figure 6: Transmitter US Oscillator Level 2 Block Diagram

(Noah Robertson)

Transmitter Circuit Simulation

An LTspice simulation schematic for the transmitter circuit is shown in Figure 7 below.

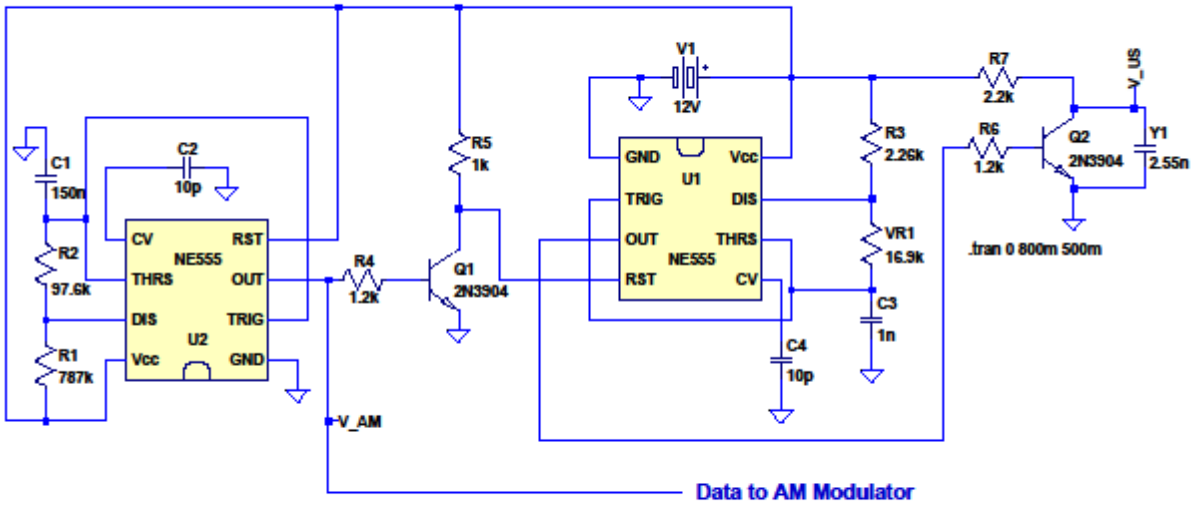


Figure 7: Transmitter LTspice Simulation Schematic

The waveforms shown in Figure 8 shows that the circuit is generating the 90% duty cycle, 10 Hz signal for the AM modulator (V_{AM}) while simultaneously turning on the oscillator for the ultrasonic transducer (V_{US}) during the radio off cycles.

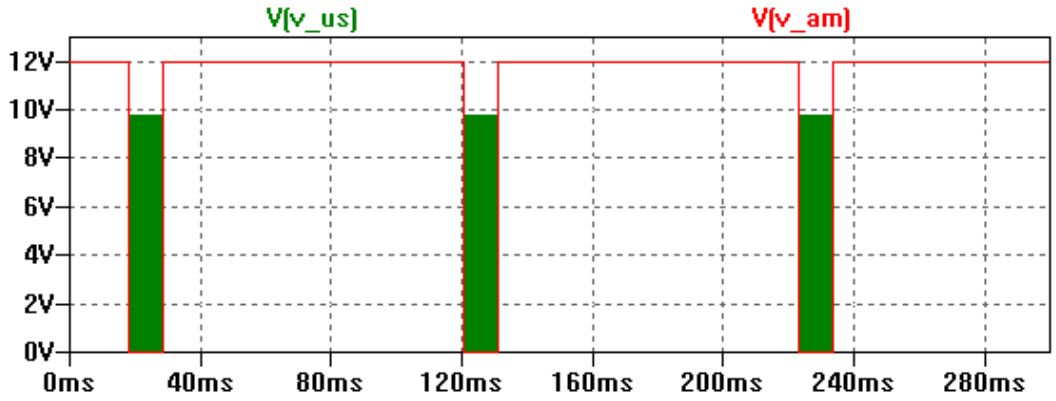


Figure 8: Transmitter LTspice simulated waveform

Transmitter Enclosure Design:

The transmitter circuit will be contained in a small enclosure which will ensure that the transmitter circuit is not easily damaged. The enclosure will have small holes in the design in order to minimize any negative effects that the enclosure may have on the ultrasound element of the transmitter. This enclosure will also allow the transmitter to be easily carried or worn. By using this design, the transmitter will be small and light enough to be worn by a child without affecting the child's ability to maneuver and chase the ball. (Robert Haver)

Transmitter Power Design:

In order to supply power to the Transmitter, the circuit in Figure 9 was used. A 12V disposable battery pack will be used as the voltage source for this circuit. This 12V source will be regulated by and LDO in order to provide a consistent 12V to the necessary electronics.

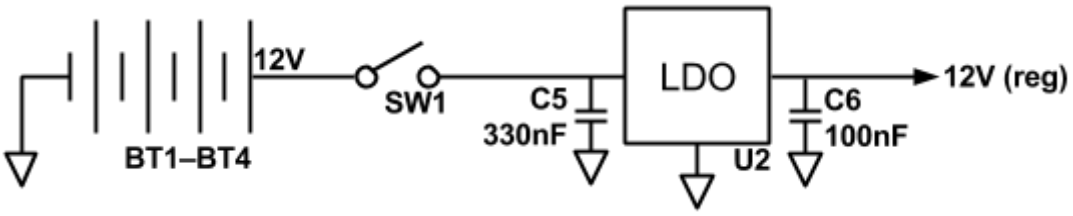


Figure 9: Transmitter Power Circuit

There were only two IC's used in this design: the 556 timer and the transmitter module. These two components consumed the majority of the power in the circuit. The 556 timer consumed a total of 360 mW when running at a full 12 volts, while the transmitting module consumed 216 mW. After totaling up the power of all the components in the transmitter, the total power was calculated to be 2 watts. The operation time of the transmitter is designed to run for one hour.

By evaluating the power of the circuit with P=VI, at 12V and 2W, the current can be found to be 167 mA. If the circuit is to be powered for one full hour, then the battery would need to be rated for at least 167 mA·H at 12V. The voltage source that was chosen to meet this stipulation consists of four disposable disc batteries that are rated for 3 volts

at 80 mA/h. These four batteries are then placed in series to create a total supply of 12 volts and to run for 240 mA/h. A 12 volt LDO is also placed within the design to help step up, or step down the voltage from the batteries. If the voltage of the supply rises, and eventually falls during consumption, the regulator will account for the change in voltage to make sure that 12 volts is consistently applied to transmitter circuit.

(Dan Madden)

Receiver Level 1 Design:

A more detailed functional description of the receiver unit can be seen in Figure 10. A range-finding unit measures time of arrival of radio and ultrasonic pulses from the transmitter to determine the distance to the child. The direction finder unit establishes a bearing from the child by sensing which of five circularly-placed directional antennas is receiving the maximum signal from the transmitter. An inertial measurement unit (IMU) senses inertial information of angular velocity and linear acceleration. The CPU uses range, bearing, and inertial information to make decisions about desired drive direction, which it transmits to the drive system, which transforms that signal to mechanical motion of the wheels. The functional requirement of each block in Figure 10 can be found in Table 6. (Noah Robertson)

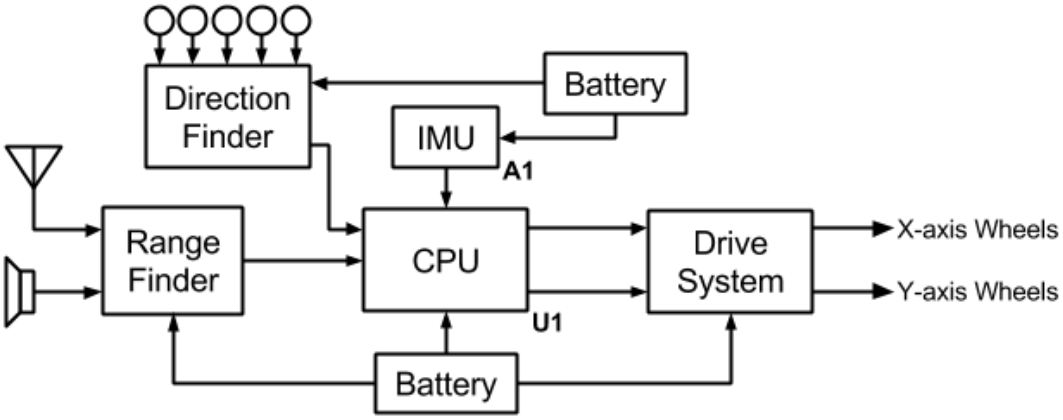


Figure 10: Receiver Level 1 Block Diagram

Receiver Drive System Level 2 Design:

The drive system converts drive commands from the microcontroller into mechanical motion of the wheels. Two parallel, identical drive systems are formed by four motors and two motor controllers as shown in Figure 11.

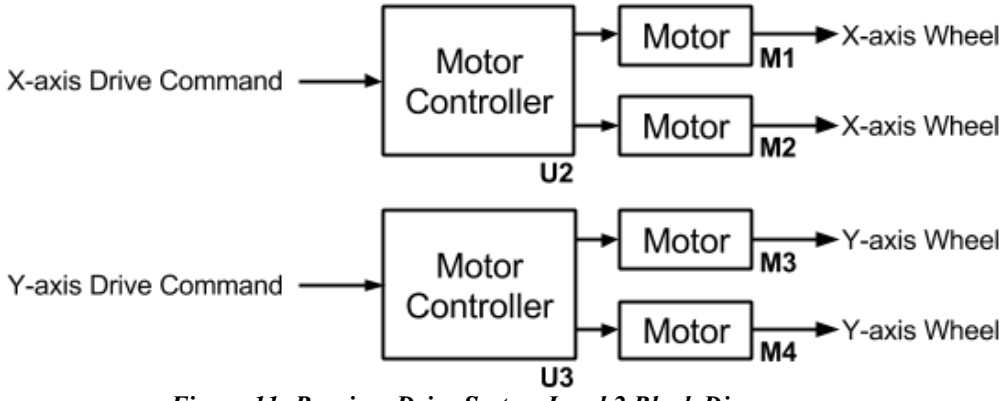


Figure 11: Receiver Drive System Level 2 Block Diagram.

The motors convert electrical power from the motor controller into mechanical power to the wheel axles, with the amount of electrical power fed into each motor regulated by the motor controllers proportionately to the level of signal commanded from the microcontroller. The functional requirement of each block in Figure 11 can be found in Table 7. (Noah Robertson)

Receiver Drive System Software Level 2 Design:

Given a situation as shown in Figure 12, with some arbitrary range to the transmitter and some arbitrary bearing (as developed from the robot y-axis as shown), the program will decompose its own inertial velocity into two vectors, one along that bearing line and one perpendicular to it.

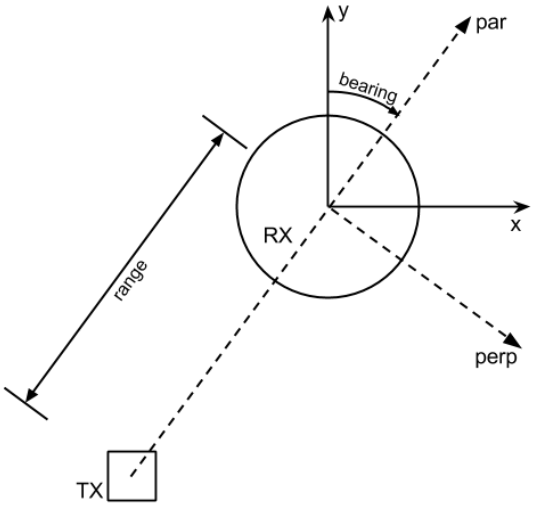


Figure 12: Drive Control Operation.

The algorithm counteracts the perpendicular velocity to halt any superfluous side-to-side motion by accelerating inversely proportionately to that velocity. A desired parallel speed is calculated as proportional to how far from a nominal distance to the transmitter the ball currently is. Based on the existing parallel speed, the algorithm will accelerate as necessary to reach the desired speed. This will accelerate the ball in a discrete series of steps to the desired speed and direction. As the ball approaches the desired velocity—including zero velocity when the child is stopped—the accelerating steps will become vanishingly small.

To prevent the robot from accelerating faster than the ball can accelerate, or to recover from external angular disturbances, angular velocity information from the IMU is pulled before each step. If that angular velocity exceeds a specified rate (i.e., when the robot begins to spin around within the ball), a fail-safe loop forces the robot to drive in the opposite direction until the angular momentum is damped to a controlled level.

It should be noted that to determine the present velocity of the ball, absolute velocity information is not necessary. Assuming that neither the robot in the ball nor the ball on the ground are slipping—which is the normal mode of operation—then the linear velocity of the wheels will exactly equal the linear velocity of the ball. This allows one to forego

translating the voltage of the signal sent to the wheels into an actual linear velocity and simply treat the voltage of the drive signal as the velocity.

The pseudo-code below in Figure 13 illustrates the algorithm used to decide the appropriate drive signal to the wheels based on range and bearing to the transmitter and inertial data from the IMU. Inputs, outputs, and tuning parameters used in this code are described in Table 2.

Table 2: Input, Output, Tuning Parameters Drive System Code.

INPUTS	**Units**	**Description**
x_roll, y_roll	rad/s	Angular velocity from IMU
bearing	degrees	Angle from null finder
range	m	Distance from internal register (from range finder program)
OUTPUTS	**Units**	**Description**
x_drive, y_drive	V	Voltage to motor controller
TUNED PARAMETERS	**Units**	**Description**
max_safe_roll_rate	rad/s	Maximum safe angular speed
drive_radius	V*s/rad	Converts angular speed to linear speed
nominal_range	m	Desired range to keep between transmitter and receiver
max_velocity	V	Defines upper bound of motor speed
max_delta_velocity	V	Defines maximum velocity step for smooth operation

```
LOOP
  WHILE ABS(x_roll) > max_safe_roll_rate OR ABS(y_roll) > max_safe_roll_rate
    x_drive = - x_roll * drive_radius
    y_drive = - y_roll * drive_radius
  END WHILE

  velocity_perp = x_drive * COS(bearing) - y_drive * SIN(bearing)
  velocity_par = x_drive * SIN(bearing) + y_drive * COS(bearing)

  delta_velocity_perp = -velocity_perp * max_delta_velocity

  range_offset = nominal_range - range
  desired_velocity_par = range_offset * max_velocity
  delta_velocity_par = (desired_velocity_par - velocity_par) * max_delta_velocity

  delta_velocity = SQRT(delta_velocity_par^2 + delta_velocity_perp^2)
  delta_velocity_direction = bearing + ATAN2(delta_velocity_perp, delta_velocity_par)
  delta_velocity = max_velocity IF delta_velocity > max_delta_velocity
  delta_velocity = -max_velocity IF delta_velocity < -max_delta_velocity

  delta_x_drive = delta_velocity * SIN(delta_velocity_direction)
  delta_y_drive = delta_velocity * COS(delta_velocity_direction)

  x_drive += delta_x_drive
  x_drive = max_velocity IF x_drive > max_velocity
  x_drive = -max_velocity IF x_drive < -max_velocity

  y_drive += delta_y_drive
  y_drive = max_velocity IF y_drive > max_velocity
  y_drive = -max_velocity IF y_drive < -max_velocity
END LOOP
```

Figure 13: Drive Control Pseudo-Code.

Receiver Drive System Software Simulations:

To test the practicality of the control software and to determine the necessary directional resolution for smooth and effective operation, the drive control algorithm was implemented in software. A simulated environment was also developed to gather information about ball response given several different strategies of user interaction. The simulation was written is ruby and is available in Appendices A-C.

Each simulation run started with the ball 2 m from the child at a random angle. The simulator moved the child at 3 m/s in 100 ms steps (the same frequency as the ranging pulses from the transmitter) according to one of seven algorithms. One algorithm (referenced hereafter as "chase") had the child simply run directly toward wherever the ball was, and one (called "evade") had the child run directly away. The third algorithm ("line") had the child travel in a straight line along a random direction; the fourth ("line_stop") had the child travel in a straight line for the first half and then come to a complete stop; the fifth ("to_from") had the child alternate running toward and away from the ball; and the sixth ("to_zig") alternated chasing the ball with running in a random direction. The final algorithm ("zig") had the child run short lines, turning to a new random direction periodically—that is, zigging and zagging. The ball followed the drive control algorithm above with nominal range set to 2 m and maximum speed at 5 m / s. The simulations represented an extremely worst-case scenario, because the child maintained absolute maximum speed over each run, and accelerated instantaneously on the turns, whereas the ball was throttled to its minimum design speed.

Each pattern ran 50 times for 12 s, plotting ball and child position in the x–y plane, as well as tracking range over time. Statistics were gathered for maximum and average speed of the ball, and minimum, maximum, and average range between the ball and the child.

As mentioned above, the simulator was also intended to determine the minimum practical number of antennas needed for the direction finder. With three antennas, the ball would have a directional resolution of 360°/3 = 120° and thus an accuracy of 120°/2 = ±60°; with four antennas, the resolution would be 90° with accuracy ±45°; et cetera. To determine this, each of the 50 simulation runs of each of the seven child behavior algorithms was run assuming three, four, five, six, and seven antennas. This resulted in 6 * 5 = 30 different algorithm/antenna configurations, and a total of 30 * 50 = 1,500 simulations. Statistics for each of the configurations are tabulated in table 3.

Table 3: Algorithm Antenna Configuration Simulation Results

Child Behavior	Number of Antennas	Average Speed (m/s)	Maximum Speed (m/s)	Minimum Range (m)	Average Range (m)	Maximum Range (m)
chase	3	3.01	4.95	0.59	1.39	2
chase	4	3	4.69	0.71	1.4	2
chase	5	3	4.57	0.76	1.4	2
chase	6	3	4.5	0.78	1.4	2

chase	7	3	4.5	0.8	1.4	2
evade	3	4.06	7.07	2	4.92	15.23
evade	4	3.51	7.06	2	3.19	7.79
evade	5	3.22	4.74	2	2.82	3.55
evade	6	3.14	4.69	2	3.41	2.76
evade	7	3.09	4.56	2	2.73	3.34
line	3	3.09	6.09	0.79	2.92	4.58
line	4	2.97	5	0.87	2.71	3.51
line	5	2.94	4.7	0.86	2.68	3.45
line	6	2.9	4.46	0.85	2.63	3.35
line	7	2.89	4.3	0.87	2.64	3.27
line_stop	3	2.14	5.97	0.76	2.61	4.46
line_stop	4	2.07	4.99	0.82	2.47	3.65
line_stop	5	2.06	4.86	0.87	2.44	3.5
line_stop	6	2.04	4.62	0.79	2.4	3.32
line_stop	7	2.03	4.54	0.9	2.41	3.32
to_from	3	3.19	6.63	0	2.9	7.15
to_from	4	2.8	7.07	0	2.39	6.41
to_from	5	2.65	6.63	0	2.26	5.71
to_from	6	2.38	6.09	0	2.06	6.62
to_from	7	2.3	6.5	0	2.03	5.83
to_zig	3	2.94	6.73	0	1.97	5.82
to_zig	4	2.6	7.07	0	1.72	5.7
to_zig	5	2.36	6.63	0	1.57	6.7
to_zig	6	2.31	6.58	0	1.56	5.71
to_zig	7	2.25	6.87	0	1.54	5.71
zig	3	3	6.13	0.13	3.08	6.63
zig	4	2.89	6.83	0.08	2.8	5.97
zig	5	2.81	6.32	0.12	2.65	4.86
zig	6	2.8	5.98	0.08	2.66	5.51
zig	7	2.78	5.89	0.24	2.66	4.92

Before even looking at individual runs, some obvious decisions can be made. With only three antennas, the maximum range of the ball is 15.23 m, half again farther than our required communications range. This alone disqualifies three antennas as a viable option. Four antennas seems to be workable, especially with further tweaking once there is a working prototype, but to provide a margin of error, five antennas were chosen to minimize cost and complexity while maximizing operability.

Looking at the range versus time plots for the algorithm where the child chases directly after the ball shows almost identical curves in each simulation run, as shown in the typical plot of Figure 14.

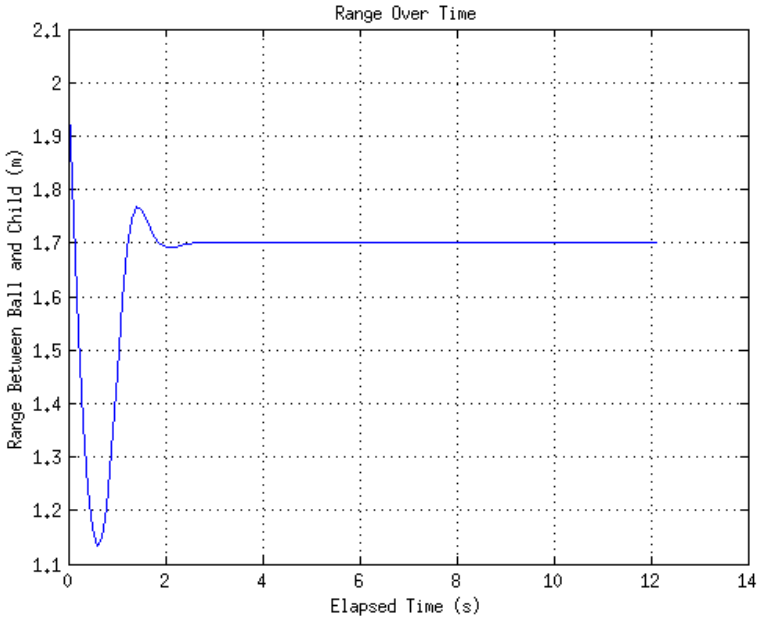


Figure 14: Drive System "chase" Simulation using 5 Antennas.

With the child running directly behind the ball at 3 m/s, the ball settles into a straight line track in one of five directions at the same 3 m/s. Since the ball is initially at rest, there is an initial dip down to about 1.2 m while it first accelerates, and then it stabilizes to maintaining a steady 1.7 m range within three seconds.

The evasion algorithm provides some more dynamic pictures. Like the chase algorithm, the plots are all practically the same, only rotated in one of five ways. A typical plot looks like the one in Figure 15. As the chase algorithm, the plots of each run are practically the same, each being only rotated in one of five ways in the x-y plane.

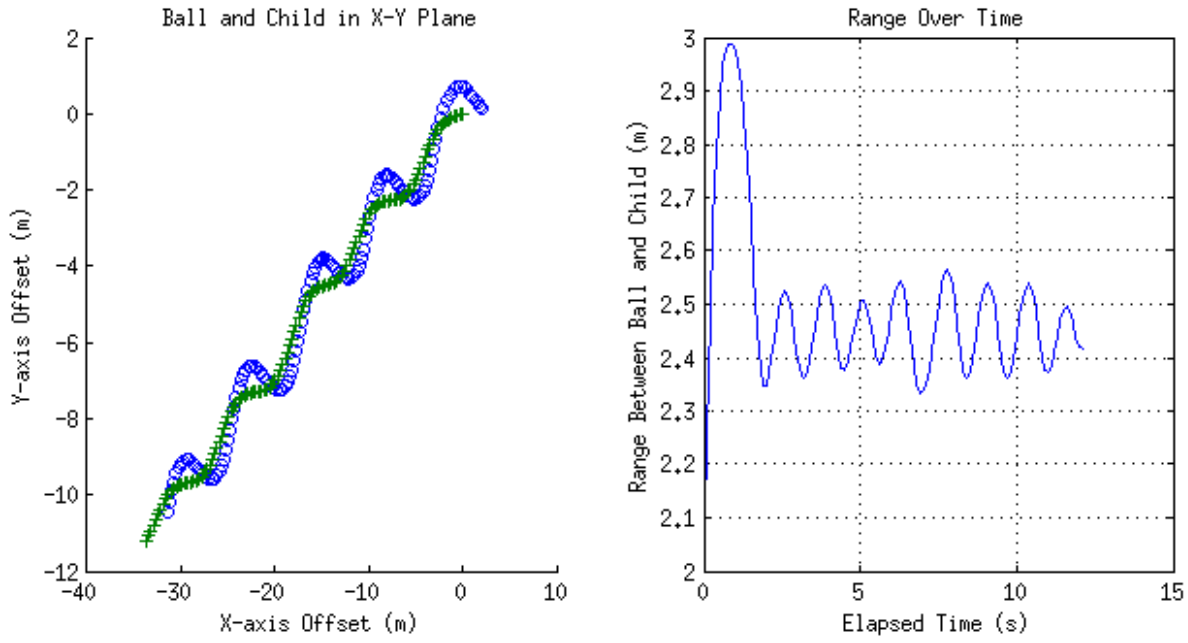


Figure 15: Drive System "evade" Simulation using 5 Antennas

The left plot is a scatter plot showing the movement of the ball (plotted with o's) and the movement of the child (marked by +'s). (To more easily interpret the chart, it should be noted that the child always starts at the origin.) As the ball chases the child, the child crosses back and forth between two of the quintants formed by the five antennas. Each time the child crosses, the ball will alter course slightly to drive towards its antenna having maximum reception. Since the simulated child always runs directly away from the ball, his course also wobbles. Looking at the right hand plot, it is clear that these deviations quickly resolve to small astable oscillations. To better illustrate why three antennas is not sufficient, one can look at the three-antenna "evade" plot in Figure 16.

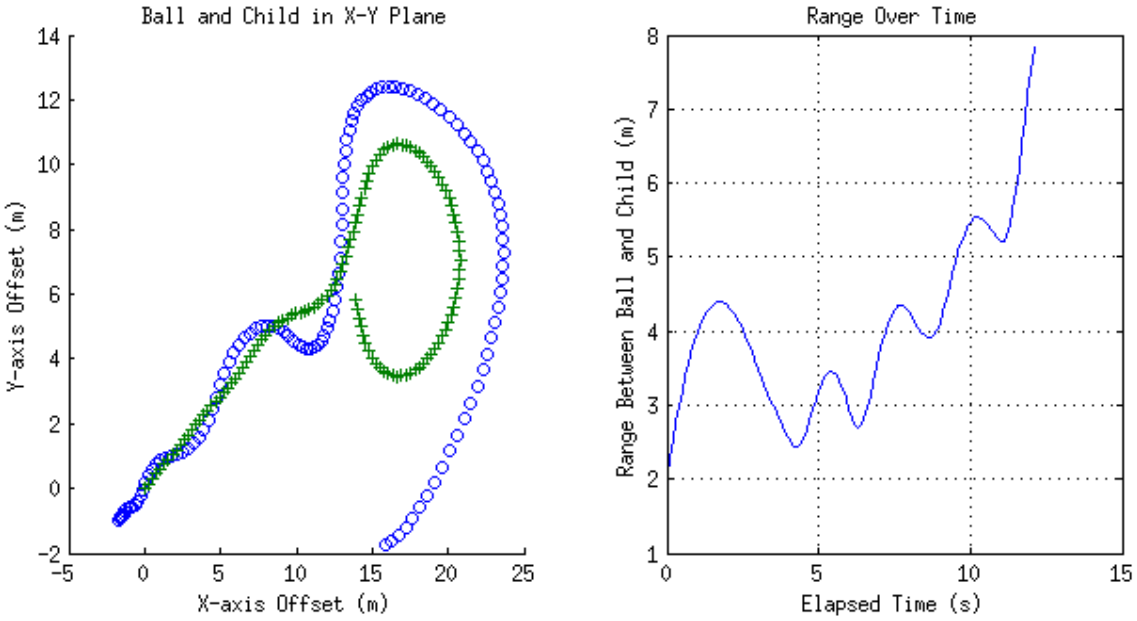


Figure 16: Drive System "evade" Simulation using 3 Antennas

Though the ball initially behaves similarly to the five-antenna design, the lower resolution proves to be insufficient to keep up with the turns of the child and the ball spirals ever farther away.

The "line" algorithm ends up being much like "evade." As seen in the representative plot of Figure 17, the ball still oscillates as the child crosses from quintant to quintant, but because the child doesn't amplify these oscillations, they become small and the range settles within five seconds at around 2.4 m behind.

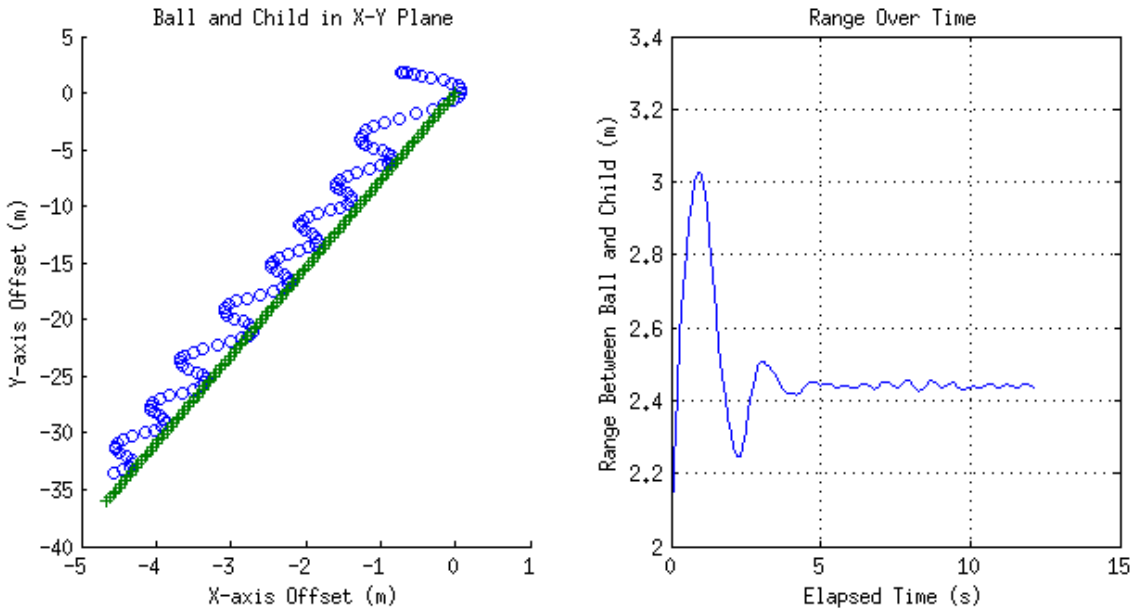


Figure 17: Drive System "line" Simulation using 5 Antennas

Figure 18 shows a variation of the "line" algorithm results. The same behavior is present, but the ball ends up maintaining range about a meter left of track. This is still acceptable behavior, because the child is running independently from the track anyway.

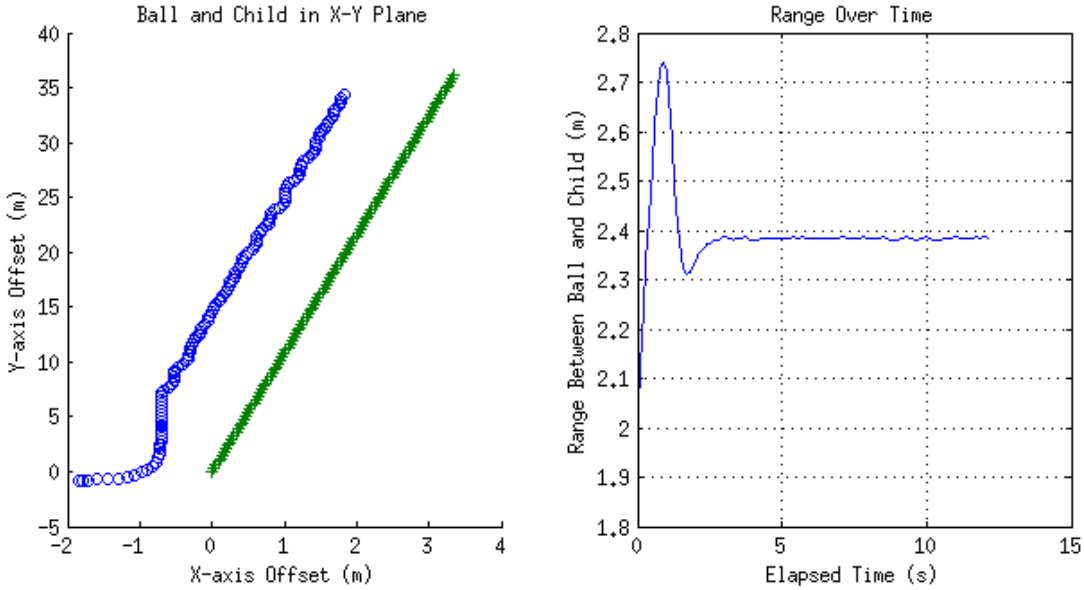


Figure 18: Drive System "line" Simulation using 5 Antennas

The "line_stop" algorithm begins identically, but when the child stops, the ball keeps rolling toward the child, getting as close as 1.5 m, then quickly (within 2 seconds) compensating and coming to a stop exactly 2 m away. The range chart for this process was nearly indistinguishable for all runs, and looks like the chart of Figure 19.

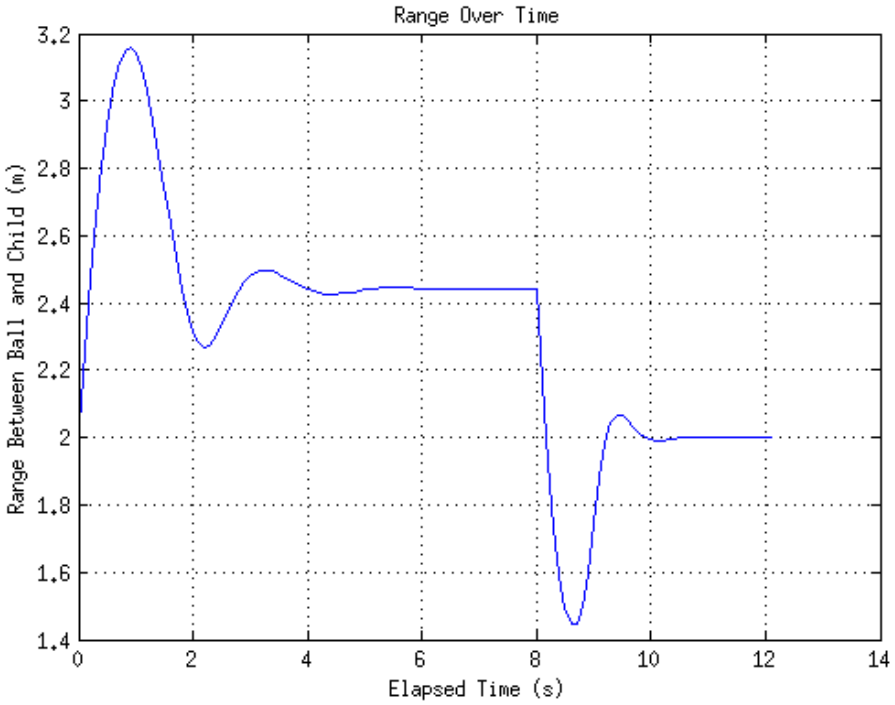
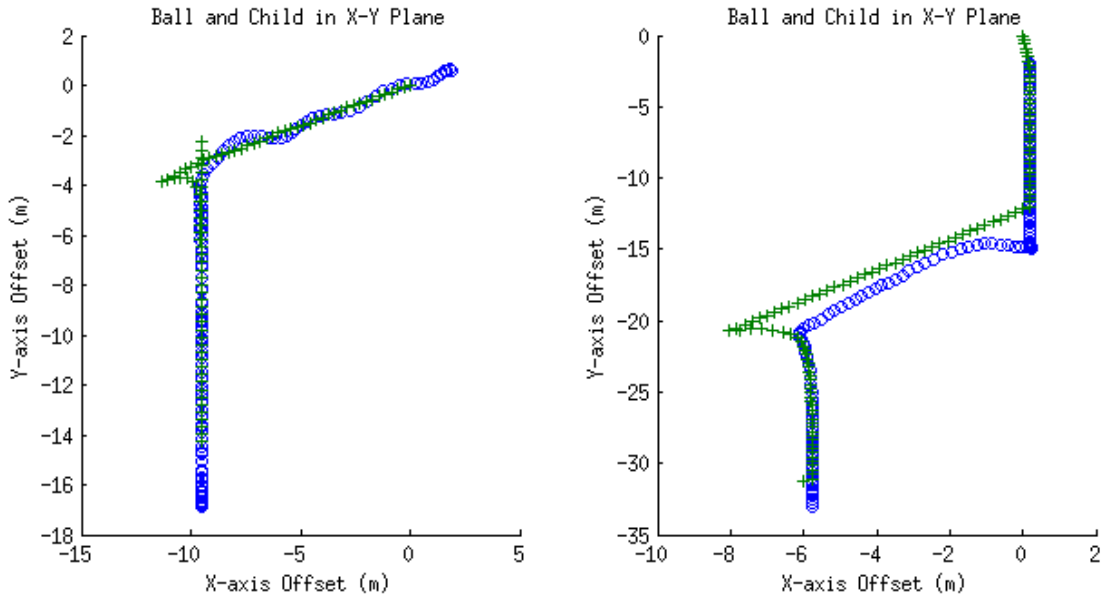


Figure 19: Drive System "line_stop" Simulation using 5 Antennas

Thus far, the ball has behaved perfectly for all simulations. Where some problems begin to creep in is with the "to_from" and "to_zig" algorithms, where the child alternates chasing and running away from the ball or running in a random direction. Successful plots of the "to_from" and "to_zig" algorithms are shown in Figure 20A and 20B.



(A) (B)

Figure 20: Drive System "to_from"(A) and "to_zig"(B) Simulation using 5 Antennas

The problems arise when the child reverses direction just as the ball is accelerating toward him/her as seen in Figure 21.

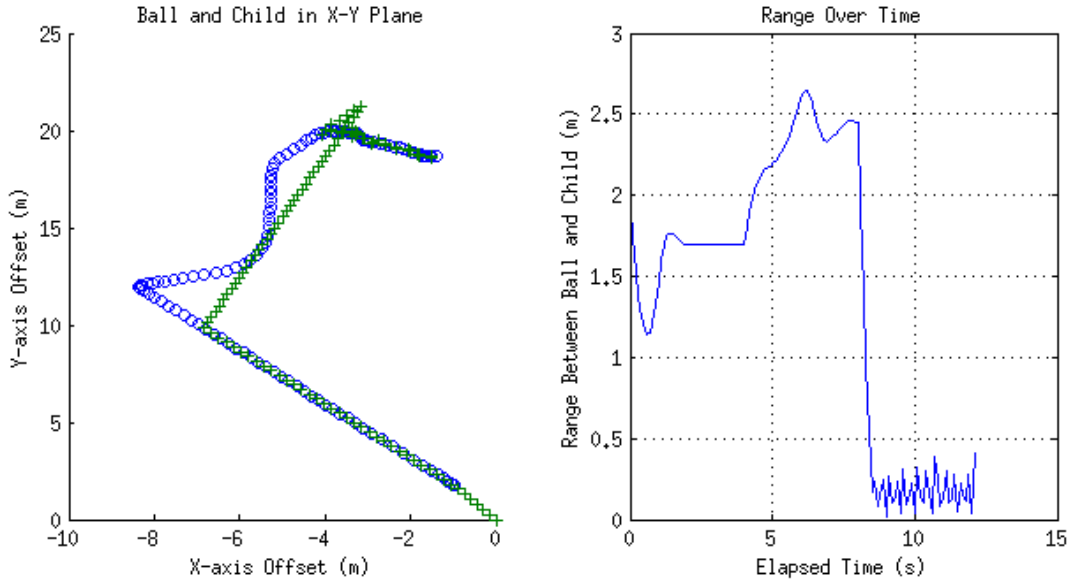


Figure 21: Drive System "to_zig" Simulation using 5 Antennas

With the child and the ball now moving full speed directly toward each other, the ball does not have enough time to reverse course and is thus caught. For the practical prototype, this situation will be mitigated somewhat because the child will not be able to accelerate infinitely (or even especially quickly) and can be corrected by increasing the acceleration capability of the ball.

Some of the most interesting plots come when the child simply runs around randomly as in the "zig" algorithm. From the best-behaved response of Figure 22, to the more dynamic response of Figure 23, the ball always manages to maintain within appropriate range of the child.

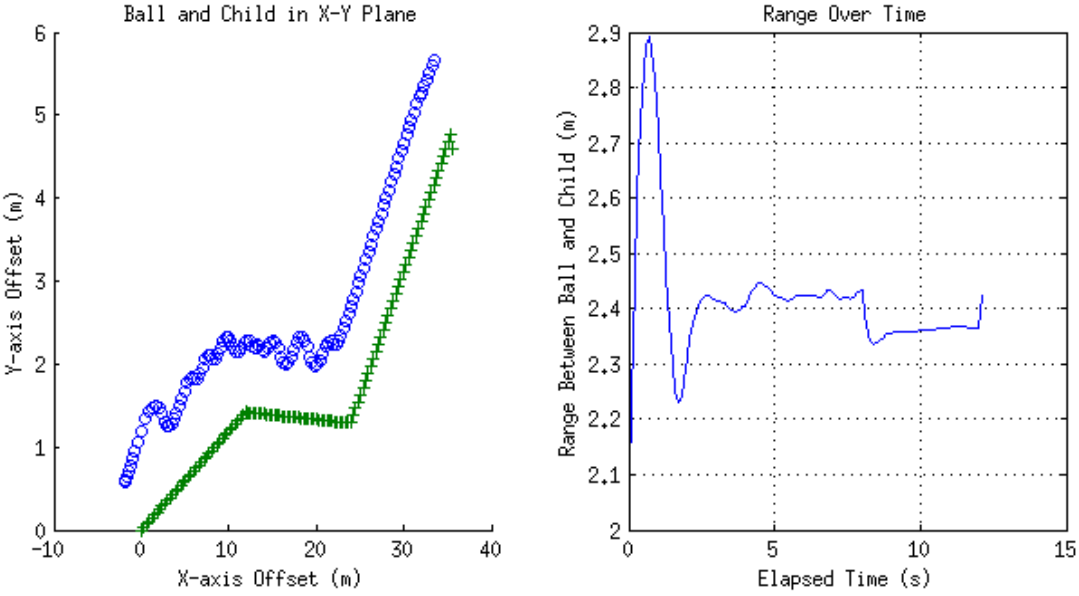


Figure 22: Drive System "zig" Simulation using 5 Antennas

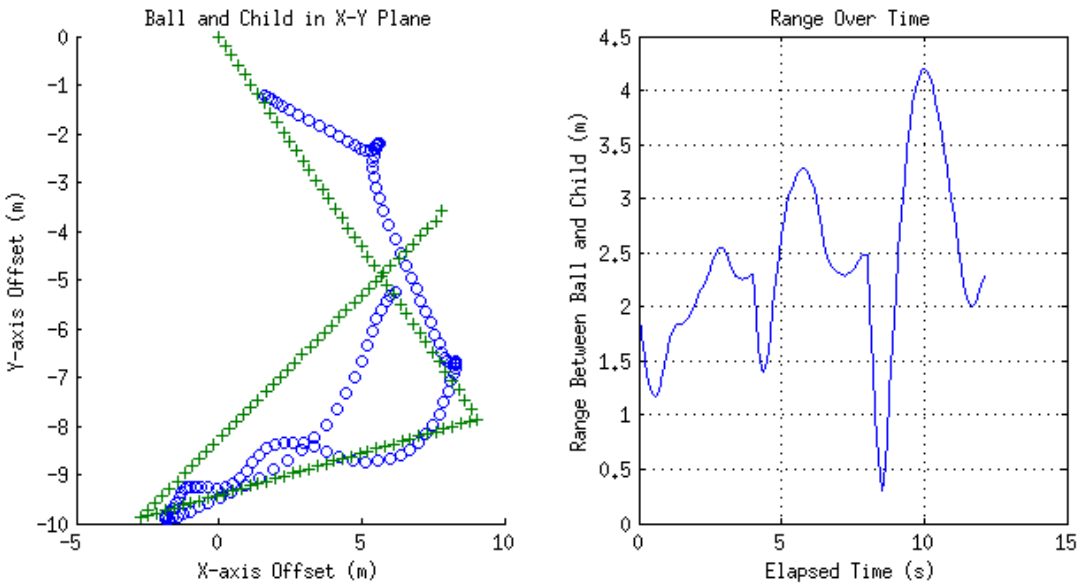


Figure 23: Drive System "zig" Simulation using 5 Antennas

Receiver Direction Finder Level 2 Design:

The first design for the direction finder used a loop antenna on a rotatable platform to establish the bearing to the transmitter. This design was replaced with five helical antennas, equally spaced in a circular pattern, which are used as an array to determine bearing. It turns out to be not necessary to resolve the bearing to such accuracy (as determined by the simulations run above), and removing the servo makes fewer moving parts in the already quite dynamic frame of the robot. The five antennas are connected electrically as shown in Figure 24. A level meter on each antenna delivers a voltage proportional to that antenna's signal strength. The voltages between adjacent level meters are compared to generate five digital bits that can be read by the microcontroller. Each bit signals if its left-hand antenna has a larger signal than its right-hand one with a digital high. (Otherwise it is low.) The microcontroller establishes the highest signal--and thus the bearing to the child--by finding the pair of bits that indicate an antenna has higher signal than both of its neighbors. The functional requirement of each block in Figure 24 can be found in Table 8. (Noah Robertson)

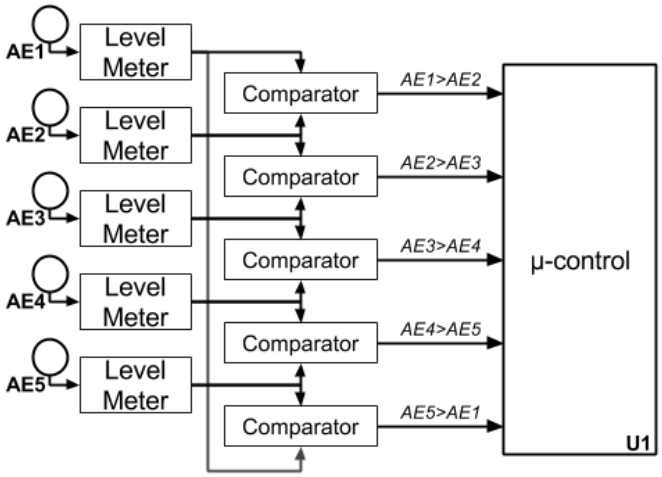


Figure 24: Receiver Direction Finder Level 2 Block Diagram.

Receiver Direction Finder Level 3 Design:

The direction finder is made up of five identical circuits each consisting of a level meter feeding into two adjacent comparators as discussed above. A schematic drawing showing detail for one of these circuits is shown in Figure 25. Each level meter is composed of an amplifier, and envelope detector, and a low-pass filter. The following discussion will focus on the single branch shown, but circuit descriptions are identical for the other four branches (with sequential reference designators).

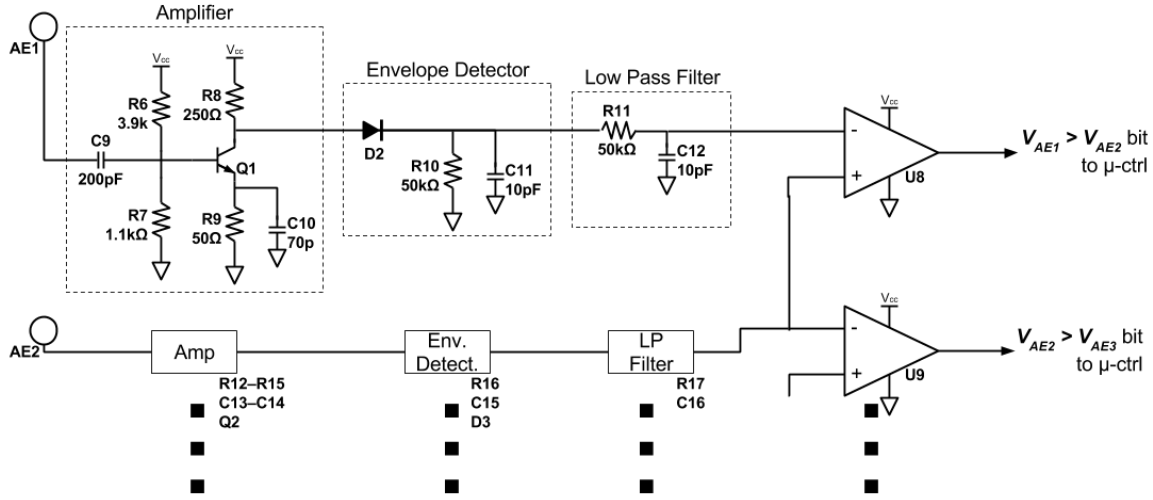


Figure 25: Receiver Direction Finder Level 3 Block Diagram

To amplify the small signal from the antenna, a high-speed NPN BJT transistor is used as a common-emitter amplifier. Blocking capacitor C9 holds the biasing DC voltage (1.1 V) at the base of Q1 from the voltage divider formed by R6 and R7. Q1 acts to amplifies its base current though its collector, thus developing a voltage across R8 proportional and greater than V_1.

The envelope detector is used as a responsive peak detector to find the maximum amplitude of the signal (plus DC offset) from the amplifier. To be effective, an envelope detector time constant must be much less than the inverse of the carrier frequency. Since the carrier signal frequency, f_c, is so high and the needed responsiveness of the system is so low (the user can't change angle to the ball anywhere near the MHz range), the time constant was selected to be $20/f_c$ = 500 ns. Standard resistor and capacitor values were selected as shown to meet this requirement.

A low-pass filter was added after the envelope detector to remove the ripple and present a more stable signal to the comparator. The filter has the same time constant as the envelope detector, yielding a bandwidth of 2 MHz (much less than the signal carrier frequency).

The comparators determine which of each pair of adjacent antennas has the larger signal, and send TTL-level binary signals to the microcontroller for processing.

(Noah Robertson)

Receiver Direction Finder Simulation:

An LTspice simulation was done for a single level meter of the RDF circuit. The schematic used is shown in Figure 26.

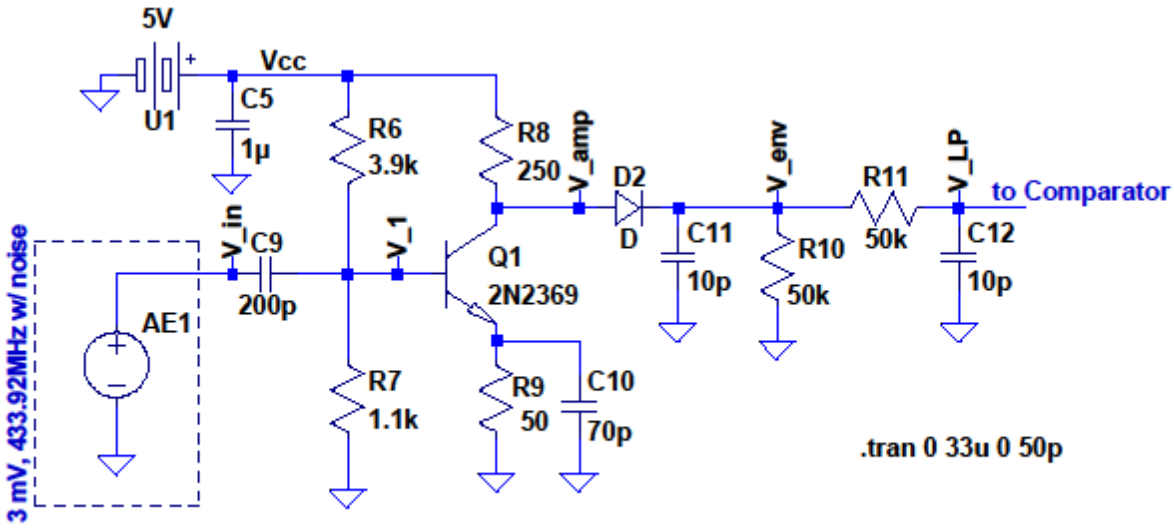


Figure 26:LTspice Single Level Meter RDF Simulation Circuit

To simulate the antenna, an arbitrary-behavior voltage source was used with the function

$$V = 0.003 * \sin(2 * \pi * 433.92e6 * t) + 0.001 * \text{white}(20e9 * t). \quad (6)$$

This generates a 3 mV sine wave at 433.92 MHz with 500 µV of noise. A close-up view of this circuit's functioning is shown in Figure 27.

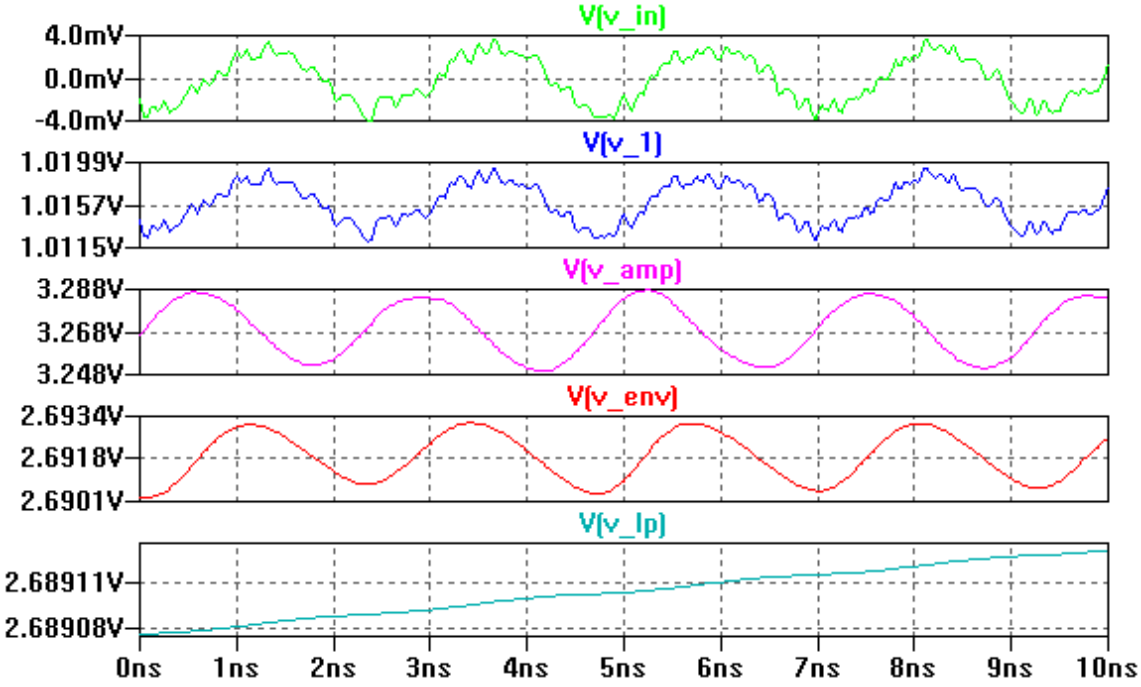


Figure 27: Waveform Simulations of RDF Simulation Circuit

As the noisy signal enters the circuit (V_{in}), it gains a DC offset through C9 as seen at V_1. The amplifier formed with Q_1 amplifies the voltage with a DC gain of just over three, and a small-signal gain of around five as shown in the third plot, V_{amp}. V_{env} shows the output of the envelope detector settled at the peak voltage less the voltage drop of the diode, and the final low-pass filter smooths out the ripple for a steady signal to the comparator.

Figure 28 shows the response from the low pass filter as the signal strength (top plot) steps up and down.

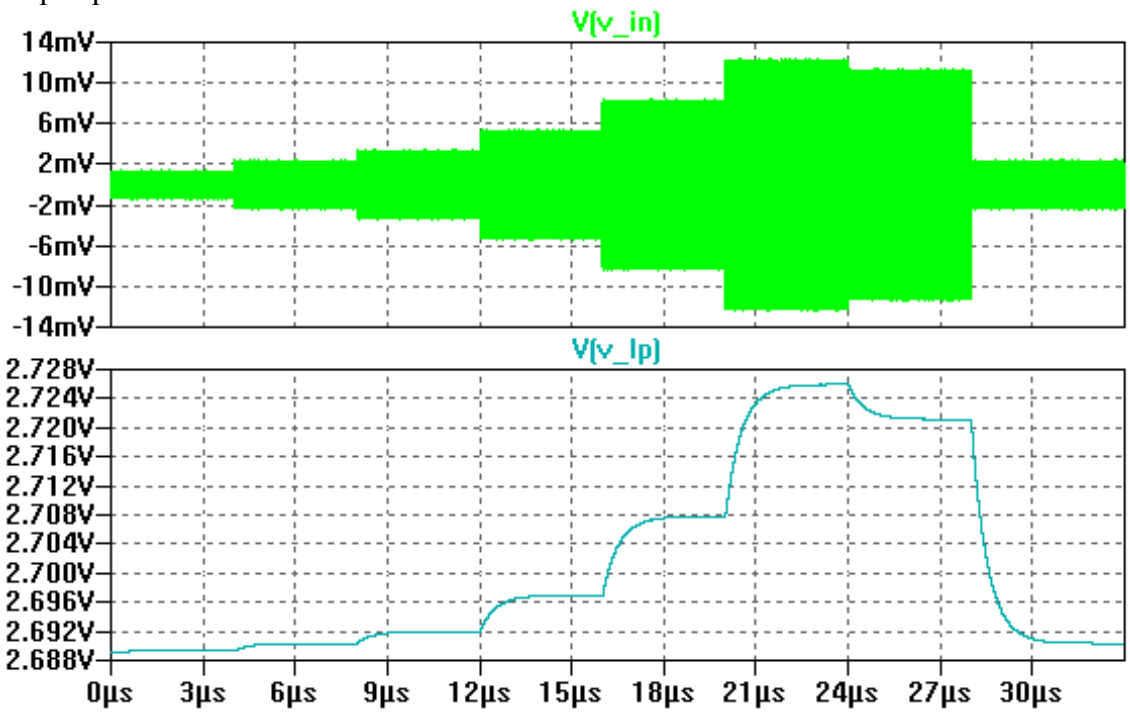


Figure 28: Low Pass Filter Response of RDF Simulation Circuit

The steps, as small as 1 mV, are clearly visible at V_{LP}. The response time as the antenna changes levels is on the order of 1 µs, far faster than a child will be able to run around the ball.

To explore the interactions between each branch and test the comparator setup, the single branch schematic was duplicated four more times and connected as shown in Figure 29.

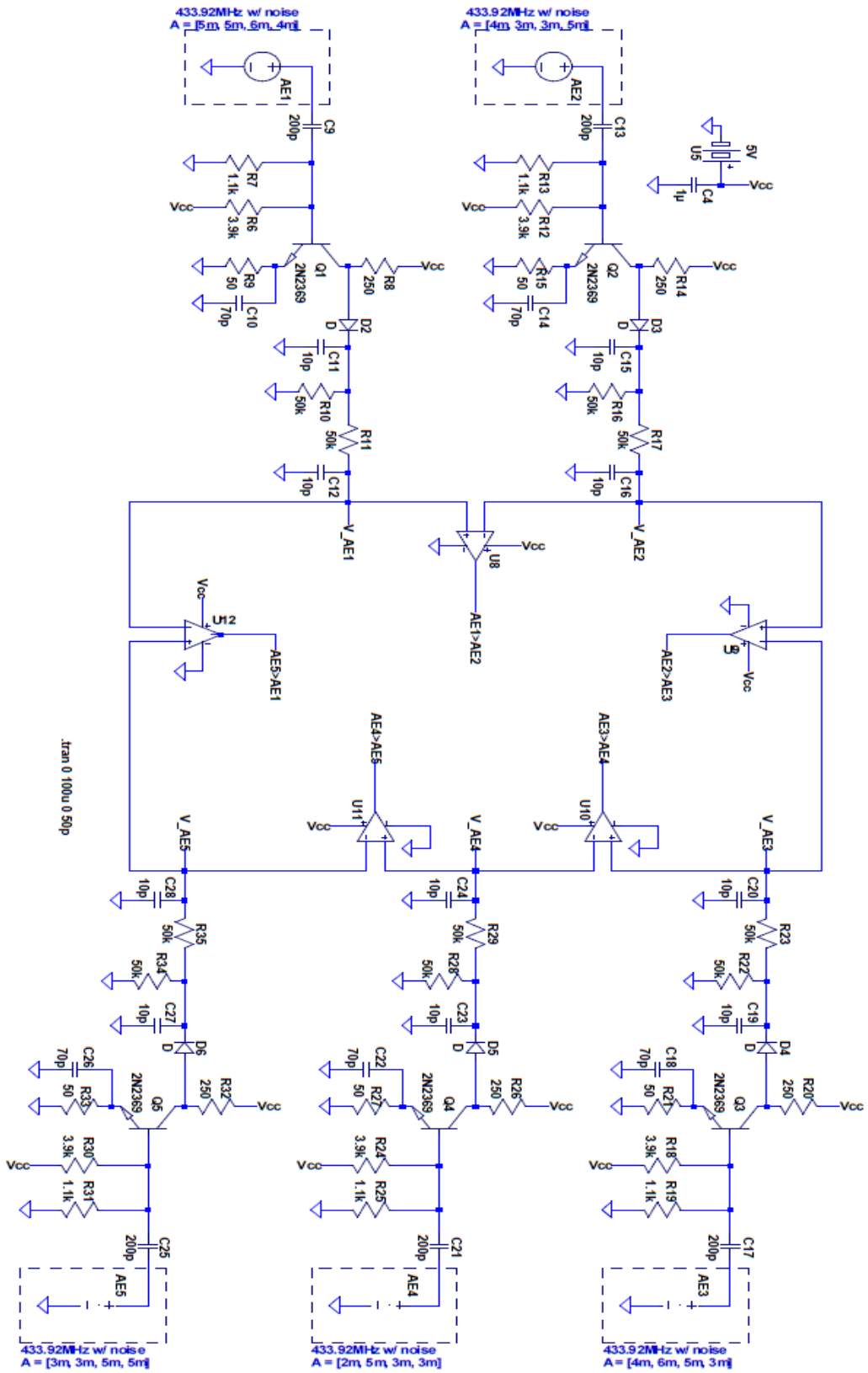


Figure 29: Complete RDF Simulation Circuit

By stepping each antenna up and down in different directions, the response of the whole circuit was found to work as designed. Figure 30 shows the output of each level meter on the top plot and the output of each comparator on the bottom.

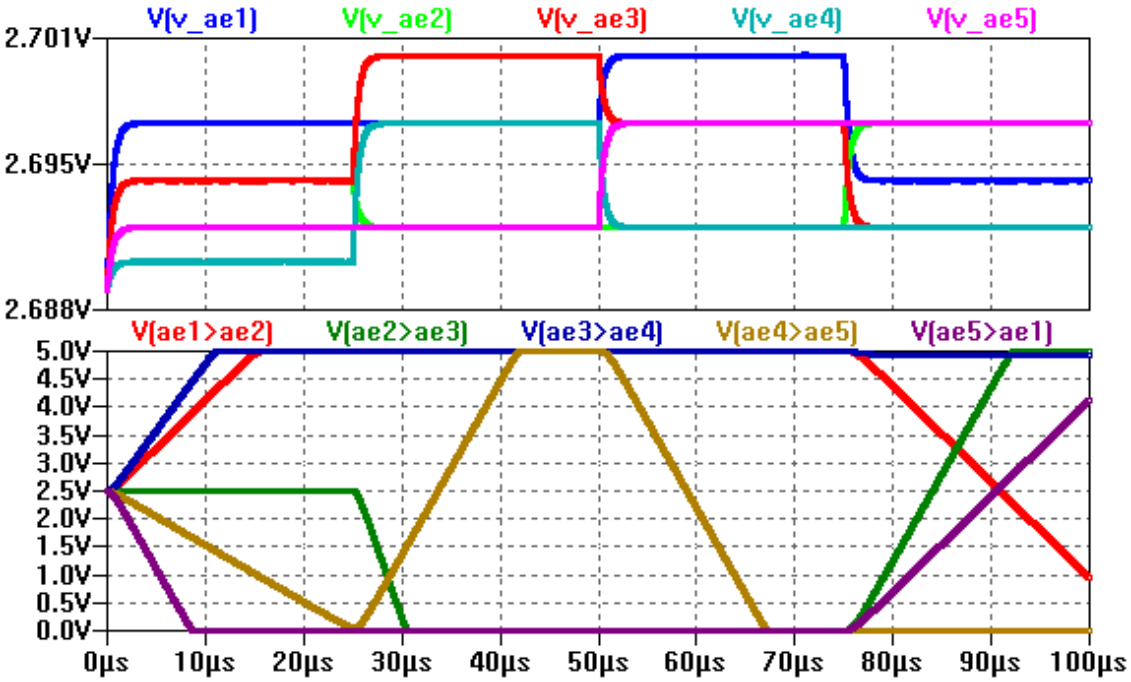


Figure 30: Response of Four RDF Simulation Circuit

The longest observed response time for the comparators is 25 µs, again much faster than the actual motion of the child will be. The binary signal from the comparators is also correct: For the first 25 µs the largest signal is received on antenna AE1, which is indicated to the microcontroller by $V_{AE1>AE2}$ being high and $V_{AE5>AE1}$ being low. The next 25 µs have AE3 with the maximum signal, and, appropriately, $V_{AE3>AE4}$ is true and $V_{AE2>AE3}$ indicates false. Similar analysis of the remaining sections show similarly accurate responses.

(Noah Robertson)

Receiver Range Finder Level 2 Design:

The range finder, shown in Figure 31, measures the difference between the time of arrival of the radio and the ultrasonic pulses from the transmitter. Envelope detection of the incoming signals creates signal pulses from which edge-triggered interrupt routines on the microcontroller are triggered. Envelope detection of the AM radio signal is accomplished with a single-unit AM demodulator and envelope detection of the ultrasound is accomplished in the US receiver circuit.

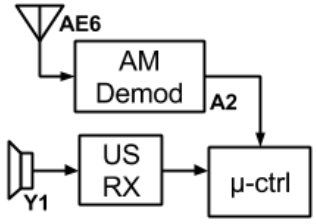


Figure 31: Receiver Range Finder Level 2 Block Diagram.

The radio interrupt is triggered on the falling edge and will simply start a timer within the microcontroller. When the ultrasonic signal will arrives tens of milliseconds later it will trigger a second interrupt routine on its rising edge which stops the timer and writes its value to an internal register for use by the drive system. The functional requirement of each block in Figure 31 can be found in Table 9.

(Noah Robertson)

Range Finder Ultrasonic Receiver Level 3 Design:

The ultrasonic receiver is a three-stage device as shown in Figure 32. The received signal passes through a 40 kHz bandpass filter to isolate the desired frequency, and envelope detector to strip out the carrier signal, and, finally, a comparator, which generates a clean step function to trigger the CPU interrupt.

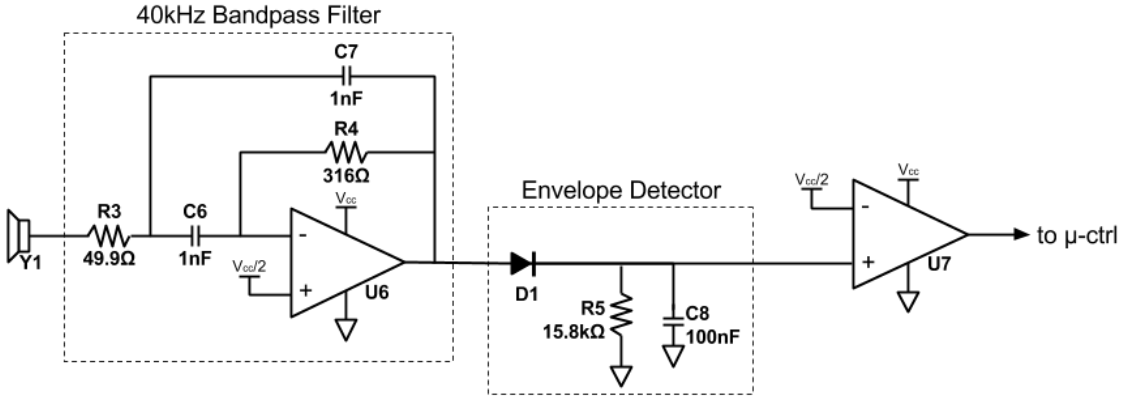


Figure 32: Ultrasonic Receiver Range Finder Level 3 Block Diagram.

The bandpass filter is a second-order filter centered at 40kHz with a 1 kHz bandwith (Q = 40). It is implemented by a Delyiannis-Friend circuit. The transfer function of this filter in standard form is

$$Ts = -1R3 * C6 s s2 + 1R41C6 + 1C7 s + 1R3 * R4 * C6 *$$

$$C7 = -K\omega 0Q s s2 + \omega 0Q s + \omega 02 \ . \quad \textbf{(Error! Bookmark not defined.)}$$

Normalizing the circuit by setting C6 = C7 = C, R3 = 1(Ω), and ω_0 = 1(Hz) implies R4 = $4Q^2$ = 6.4 kΩ and C = *12Q*= 12.5 mF. Freqency scaling by factor $K_f = 2\pi * 433.92 * 10^6$ and picking a convenient capacitor size of 1 nF sets the required magnitude scaling factor K_m as

$$C = 1 \text{ nF} = 1 \text{ F } (2 * Q * Km * Kf) \quad Km = 1 \text{ F } (2 * 40 * 2\pi * 433.92e6 * 1 \text{ nF}) = 49.736 \ . \quad \textbf{(Error! Bookmark not defined.)}$$

Scaling R3 and R4 by this factor gives the values shown in the figure (rounded to the nearest standard value).

The envelope detector needs to separate the signal (f_s = 10 Hz) from the carrier (f_c = 40 kHz). To that end, the RC time constant τ of the components should follow the relation

$$1\ fs \gg \tau \gg 1\ fc \quad 110\ \text{Hz} \gg \tau \gg 140\ \text{Hz}.$$

(**Error! Bookmark not defined.**)

To satisfy both of those requirements, the geometric mean was found between 1 fs and 1 fc at 1.58 ms. Standard resistor and capacitor values were chosen to meet this requirement.

The comparator uses a single power-supply and interfaces with a TTL-level input of the microcontroller directly to provide a clean rising edge when the ultrasonic pulse causes C8 to charge above 2.5 V.

(Noah Robertson)

Range Finder Ultrasonic Receiver Simulation:

The ultrasonic receiving circuit was simulated in LTspice using the schematic shown in Figure 33.

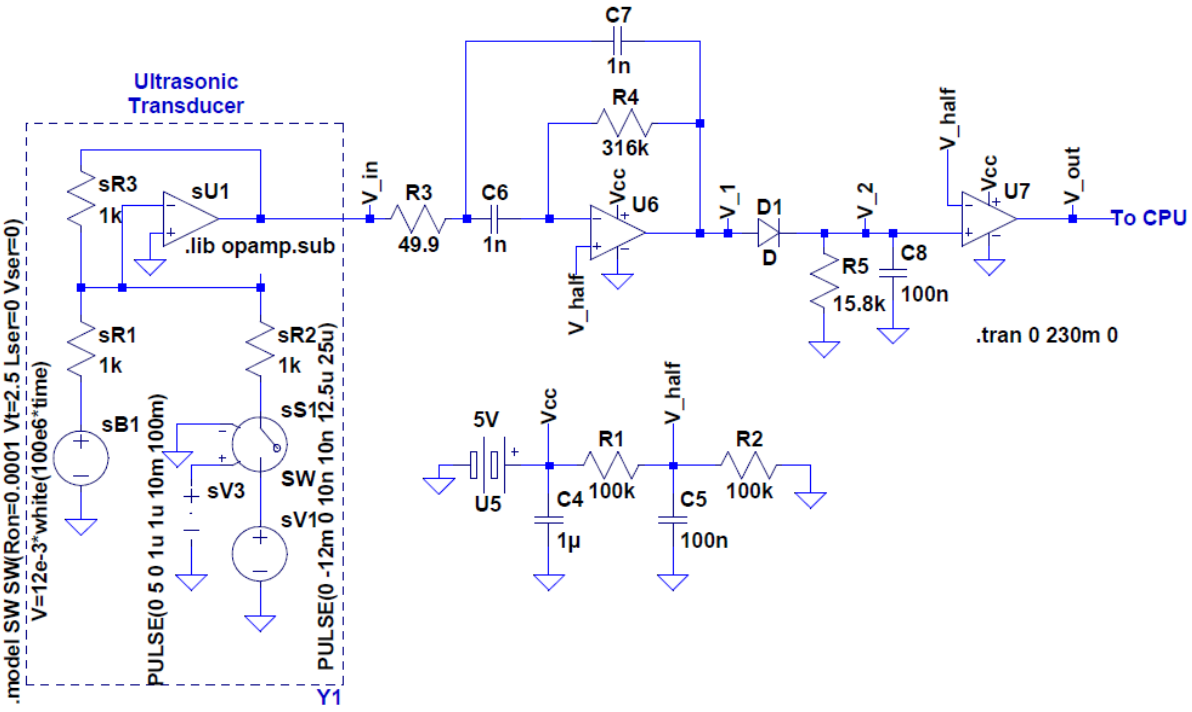


Figure 33: LTspice Ultrasonic Receiving Simulation Circuit

To simulate pulsed output of the receiving ultrasonic transducer, a 12 mV, 40 kHz pulsed signal was gated by a 10% duty cycle, 10 Hz signal. Noise was added by summing into that a white noise function with amplitude of 6 mV. Figure 34 shows this simulated noisy signal (the V_{in} to the receiver) on the top plot. V_1 shows the output of the bandpass filter stage, with the signal significantly amplified, shifted to a DC level of 2.5V, and all signals except the fundamental 40 kHz signal stripped out.

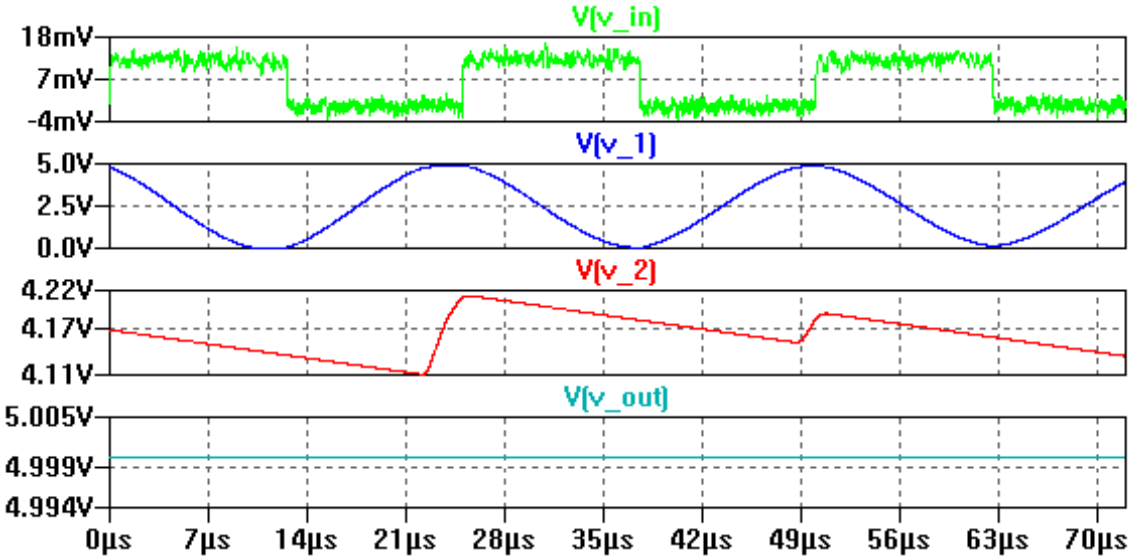


Figure 34: Ultrasonic Receiving Circuit Simulated Signal

The third plot is the output of the envelope detector. The DC level is around 4.15 V (the signal envelope less the about 700 mV drop across the diode). There is some small ripple from the envelope detector (<2%), but that is smoothed out in the comparator which outputs a steady 5V TTL high. Observing the same waveforms during the "off" cycle (as seen in Figure 35) shows V_{in} is only the noise on the line.

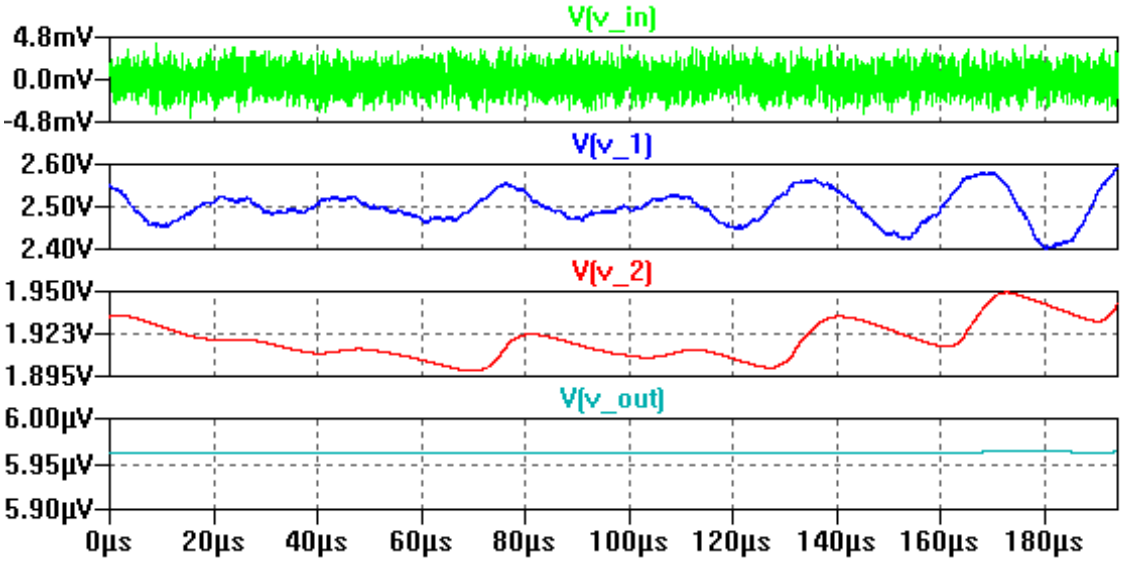


Figure 35: Ultrasonic Receiving Circuit Envelope Detector Output

There is some 40 kHz signals in the noise, but, even amplified by the active filter, it barely register at V_1, and with the voltage drop from the envelope detector diode, it remains well below 2.5 V, resulting in a solid 0V TTL low to the controller.

Looking at the simulation on a longer timescale reveals the behavior of the 10 Hz signal pulse. Figure 36 shows the rising and falling edge of the data packet.

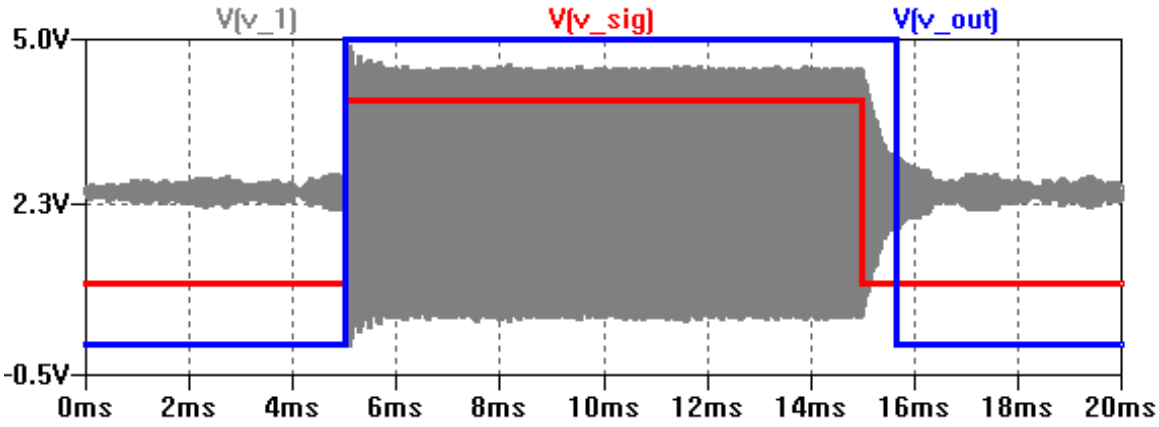


Figure 36: Ultrasonic Receiving Circuit Rising and Falling Edge

V_1 shows the output of the active filter, and V_{sig} shows the actual arrival time of the packet. V_{out} shows the output of the system to the microcontroller. On the millisecond scale, the delay between V_{sig} and V_{out} is insignificantly small, meaning that the output rising edge accurately marks the pulse arrival. (For references, a processing delay of 1/10th of a millisecond would yield only a 3.4 cm error in range.) The falling edge has a perceptible delay of about half a millisecond while the capacitors discharge, but this is inconsequential because the falling edge carries no data and the comparator switches low well before the arrival of the next pulse 90 ms latter. (Noah Robertson)

Receiver Power Design:

In order to supply power to the receiver, the circuit in Figure 37 was used. A 12V rechargeable battery pack will be used as the source for this circuit. The 12V source will be regulated by two LDO's in order to provide 12V and 5V supplies to the receiver circuit, and the regulated 5V will be applied to a voltage divider in order to get the required 2.5V supply. (Robert Haver)

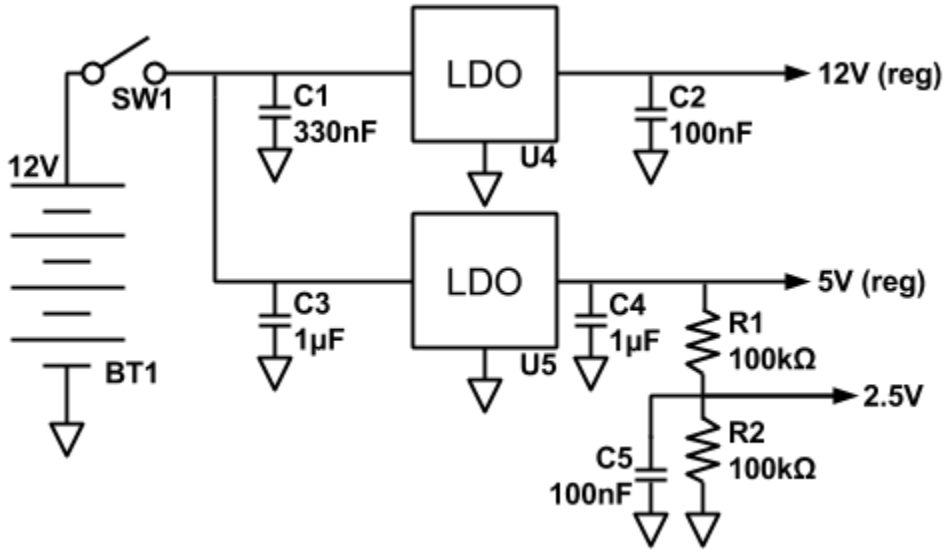


Figure 37: Receiver Power Circuit

The motors of the robot were assumed to be the main consumption of power in the robot allowing the power calculations to be based off of the motor power consumption as most other loads a negligible. The battery capacity required for our robot to run at least 1 hour at the current draw of 792 mA can be calculated by the following equation:

$$Battery\ Capacity mA \cdot h = Running\ Time hrs * Operating\ Current mA \quad (7)$$

$$Battery\ Capacity 1\ hour 1 * 792 = 792 mA \cdot h . \quad (8)$$

A 12V battery pack was thus selected with a capacity exceeding 729mA·h.

(Melissa Haver)

Mechanical Design:

The mechanical layout of the main controller is shown in Figure 38. The controller platform is propelled by two orthogonal sets of omni-wheels.

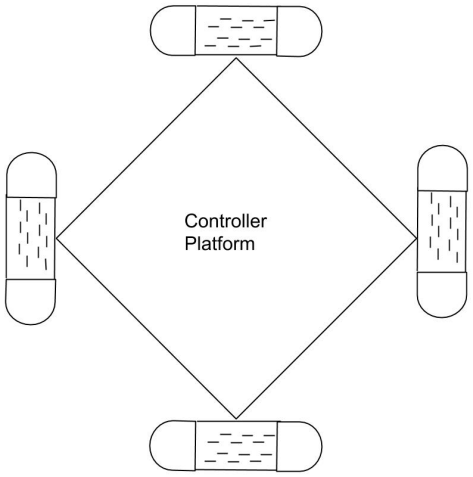


Figure 38: Controller Platform

Omni-wheels allow free movement perpendicular to the drive direction using roller bearings. This means that rather than needing steering wheels, full two-dimensional motion can be realized as the vector sum of the individual components. This platform will be placed loose inside the ball with spring-loaded stabilizers that extend vertically from the controller platform to the top of the ball. This design will maintain pressure on the wheels of the robot to increase the grip, and will prevent the robot from landing on its back inside the ball. The stabilizers consist of ball transfer casters mounted on miniature shock absorbers. These ball transfer casters allow the stabilizers to move freely along the surface of the shell as the robot moves inside the ball. The shock absorbers are attached to a platform of the internal robot, and can be adjusted so that the desired pressure is applied to the spherical shell. (Robert Haver)

Mechanical Motion:

Forward motion of the sphere will be created by the robot driving up the inside of the ball, with the weight of the robot forcing the ball forward. As the acceleration of the sphere depends entirely on the weight of the robot, the weight will have to be large enough to overcome the inertia of the ball for the sphere to move efficiently. The basic concept of this forward motion of the sphere is illustrated in Figure 39 below.

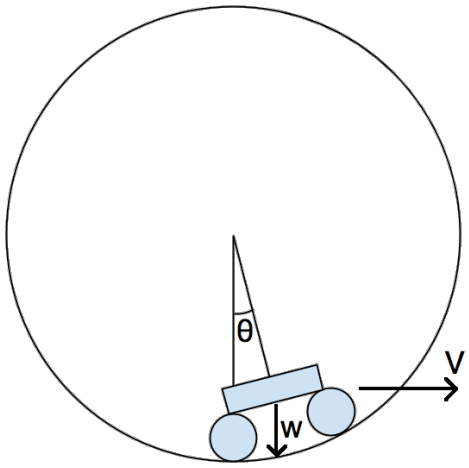


Figure 39: Mechanical Motion.

For the sphere to maintain a constant acceleration, the robot will maintain a constant angular position inside the sphere. The only component of the robot's weight that will affect the velocity of the sphere is the component tangential to the inside surface of the sphere. This tangential component of the weight, w_T, can be expressed in terms of the weight and angle θ as

$$w_T = w\sin\theta\,. \tag{9}$$

Assuming negligible or no slip between the wheels of the robot and the surface of the sphere, the velocity of the robot will be equal to the velocity of sphere.

Moments of Inertia:

The moment of inertia of a hollow sphere is

$$I_s = \frac{2}{3} m_s r^2\,. \tag{10}$$

To encourage efficient motion, the mass of the internal robot should be much larger than that of the sphere. As such, the force required to overcome the inertia of the sphere in order to accelerate from rest becomes negligible. The primary concern in terms of inertia is then that of the actual robot. For purposes of design, the robot can be considered to be a point mass rotating about the center of the sphere as shown in Figure 40 below.

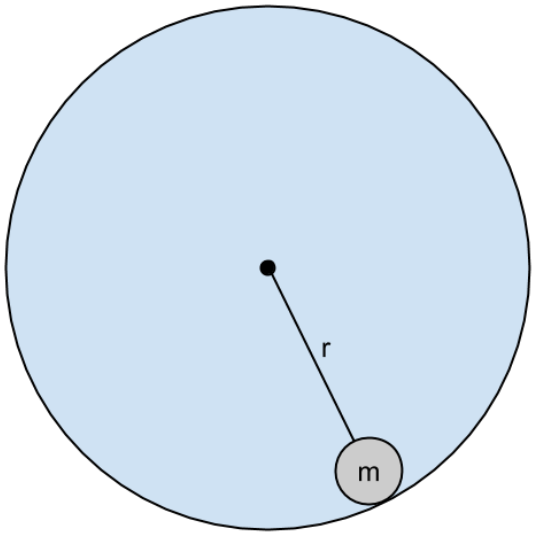


Figure 40: Robot as a Point Mass.

The only moment of inertia that then must be considered is the point mass, which is calculated using the equation

$$I_r = m_r r^2 \tag{11}$$

Acceleration Mechanics:

Assuming the system behaves as a point mass as described above, the acceleration can be determined by

$$a_s = g \sin\theta. \tag{12}$$

The maximum needed acceleration of the sphere can then be calculated by analyzing an extreme scenario, where a child instantaneously accelerates to an estimated top speed of 3m/s towards the stationary sphere positioned 2 meters away. This estimated top speed of the child was calculated through timing a 3-year-old child. In this case, the ball should be capable of accelerating to the child's top speed by the time the child catches up to it, at which point the sphere will continue to accelerate and begin to gain ground on the child. This scenario takes in to account the following conditions:

Initial Positions:	Initial Velocities:	Initial Accelerations:
$r_{0,child} = 0$	$v_{child} = 3m/s$	$a_{child} = 0$
$r_{0,ball} = 2m$	$v_{0,ball} = 0$	$a_{ball} = const.$

When Child Catches Up

$$r'_{child} = r'_{ball}$$

$$v'_{ball} = v_{child}$$

The necessary acceleration of the sphere can then be calculated by substituting these initial values into the motion equations as shown in equations 17-19 below.

$$r'_{child} = (v_{child})(t) \tag{13}$$

$$r'_{ball} = r_{0,ball} + \frac{1}{2}(a_{ball})(t^2) \tag{14}$$

$$v'_{ball} = (a_{ball})(t) \Rightarrow t = \frac{v'_{ball}}{a_{ball}} \tag{15}$$

At the moment when the child just reaches the ball (before the ball begins to pull away), $r'_{child} = r'_{ball}$ and the above set of equations can be solved to give the required acceleration of the ball as

$$a_{ball} = 2.25\frac{m}{s^2}. \tag{16}$$

Solving for θ in Equation 16 gives the maximum expected angle of

$$\theta = 13.3^\circ. \tag{17}$$

As this is a relatively small angle, the prior assumption that the robot will experience negligible slip within the sphere can be considered to be a reasonable assumption.

(Robert Haver, Noah Robertson)

Motor Calculations:

The robot will be equipped with four two inch diameter omni-wheels each attached to their own motor. By connecting each individual wheel to a motor, this gives the robot extra control while achieving the necessary speeds requirements. To calculate the Torque, τ, for each motor the forces that act on the wheel, as shown in Figure 41, must be considered.

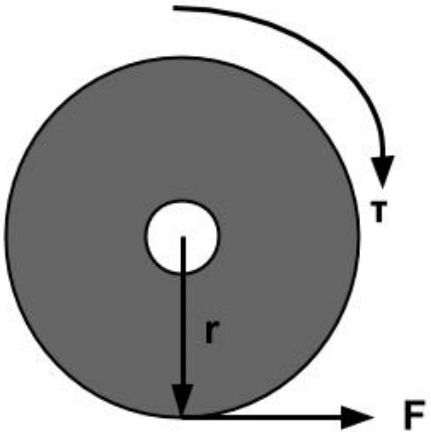


Figure 41: Wheel Free- Body Diagram

The force, F, is calculated by taking into account the total weight of the robot, the number of drive wheels, *n*, and the desired acceleration which was found in equation 20. To calculate the force, F, the weight of the robot was assumed to be 4lbs, which is equivalent to 1.814 kg, with 4 drive wheels resulting in the following equation:

$$F=Maball n=1.814*2.254=1.02\ Newtons\ . \quad (18)$$

The Torque, τ, for each motor is then calculated to find the required Torque necessary to overcome the 1.02 Newtons acting on each wheel. This can be achieved by the following calculation, where the radius of the wheel is 1 inch or 0.0254 meters:

$$\tau=F*r=1.02*0.0254=0.026\ N\cdot m\ . \quad (19)$$

The ball also has a requirement for it to be capable of achieving a top speed of 5 meters per second. To properly select the correct motors this speed is taken into consideration through calculation the revolutions per minute. Revolutions per minute, RPM, can be calculated simply through taking the velocity,$v=5ms$, divided by the wheel circumference, $c=\pi*0.051m=0.16m$, as shown in the equation below:

$$RPM=5\ m/s0.16m=1875. \quad (20)$$

Based on these calculations a 12v Brushed DC motor (part number: PAN14EE12AA1 from digikey) will be used to meet the design requirements.

(Melissa Haver)

Motor Control:

An H bridge circuit is an electronic circuit that allows a voltage to be applied across a load in either direction. This is the application which will be implemented to allow the robots DC brushed motors to run forwards and backwards. The integrated circuit part selected to use is the Toshiba TB6561NG a dual bridge driver IC. This device will also allow communication between the microcontroller and the motors as shown in Figure 42 from Toshiba TB6561NG data sheet. Since four motors will be used for this robot it will require two of these devices. (Melissa Haver)

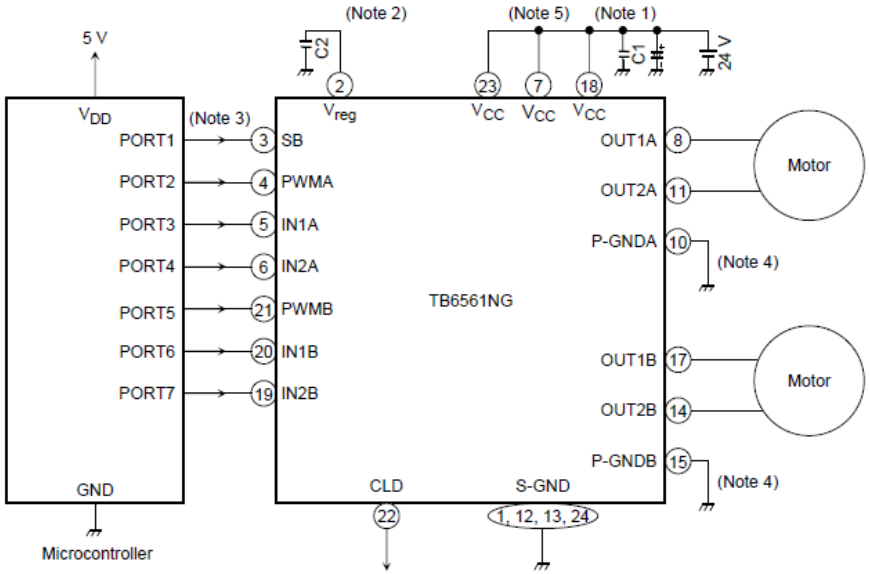


Figure 42: TB6561NG dual bridge driver IC

Robot Spherical Enclosure:

The spherical shell for the robot will be creating using a 3D printer, and will have a diameter of 10-12 inches. The design, shown in Figure 43 below, will consist of two hemispheres that will interlock to create a smooth and solid enclosure for the robot. This will allow for the sphere to be disassembled with relative ease in order access the interior components. The sphere will be printed using an acrylonitrile butadiene styrene (ABS) filament, which is a thermoplastic polymer used in Lego bricks. The ABS material will ensure that the sphere is strong and durable, while remaining relatively lightweight.

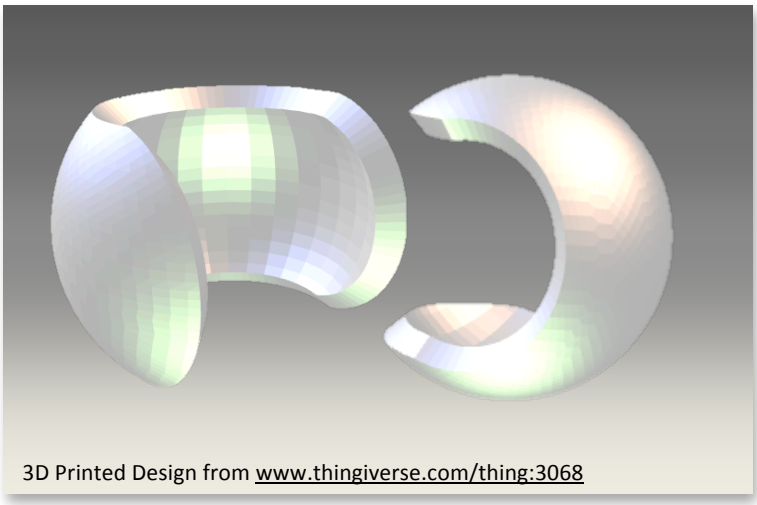


Figure 43: 3D Printed Robot Spherical Enclosure Design

5. FUNCTIONAL REQUIREMENTS

Table 4: Level 0 Functional Requirement Table

Module	**Controller**
Inputs	-received radio signal from directional antenna array (?mV) -received radio signal from omnidirectional antenna (?mV) -received ultrasonic signal from US transducer (?mV)
Outputs	Power to x-axis and y-axis wheels
Function	Interprets information from the radio antennas and ultrasonic transducer to determine relative bearing and range to the transmitter. Based on this and information of own heading and speed, makes decisions to control the drive wheels.
Module	**Transmitter**
Inputs	None
Outputs	-omnidirectional radio signal -omnidirectional ultrasonic signal
Function	Broadcasts a signal which can be used by the primary controller to locate the relative location of the transmitter.

(Noah Robertson)

Table 5: Transmitter Level 1 Functional Requirement Table

Module	**Timing Controller**
Inputs	Power from battery.
Outputs	0-12V, 10 Hz, 90% duty cycle pulsed control signal.
Function	Generates timing signal to coordinate radio and ultrasonic transmit pulses.
Module	**US Oscillator**
Inputs	-Power from battery. -Control signal from timing controller.
Outputs	Signal to drive ultrasonic transducer.
Function	Converts timing signal into discrete pulses of 40 kHz ultrasound.
Module	**AM Transmitter (A1) —— part #QAM-TX2-433**
Inputs	-Power from battery. -Control signal from timing controller.
Outputs	AM modulated signal to omnidirectional antenna.
Function	Modulates the timing control signal.

Module	Battery
Inputs	none
Outputs	12V to all electronics.
Function	Provides power to all necessary equipment. Must provide 2W of power for 1hr of operation.

(Noah Robertson/Melissa Haver)

Table 6: Receiver Level 1 Functional Requirement Table

Module	CPU (U1) —— part#PIC16F716
Inputs	-Power from battery. -Range information from range finder. -Bearing information from direction finder. -Angular velocity from IMU. -Linear acceleration from IMU.
Outputs	Drive signal to x- and y-axis drive controllers.
Function	Based on range, bearing, and inertial information makes decisions to control the drive wheels.
Module	**Range Finder**
Inputs	-Power from battery. -Signal from omnidirectional radio antenna. -Signal from US transducer.
Outputs	Range Information.
Function	Measures time of arrival difference between omnidirectional radio antenna and ultrasonic transducer to determine range to transmitter.
Module	**Direction Finder**
Inputs	-Power from battery. -Signal from directional radio antenna array.
Outputs	Bearing information.
Function	Generates parallel digital signal indicating bearing to user.
Module	**IMU(A1) —— part #SEN-10121**
Inputs	Power from battery.
Outputs	-2-axis angular velocity information. -2-axis linear acceleration information.
Function	Senses motion of the controller platform and transmits these data in a convenient form to the microcontroller.

Module	Drive System
Inputs	-Power from battery. -Drive signal.
Outputs	Power to wheels.
Function	Converts drive signal from CPU to mechanical motion of wheels.
Module	**Battery**
Inputs	None
Outputs	2.5V, 5V, 12V to all electronics and motors
Function	Must provide 9.5W of power to all necessary equipment for 1 hr of operation.

(Noah Robertson/Melissa Haver)

Table 7: Receiver Drive System Level 2 Functional Requirement Table.

Module	Motor Controller(U2,U3) ——part#TB6561NG
Inputs	-Power from battery. -Drive signal from microcontroller.
Outputs	Power to motor.
Function	Powers motor according to the commanded signal from the microcontroller.
Module	**Motor(M1,M2) ——part#PAN14**
Inputs	Power from motor controller
Outputs	Power to wheels.
Function	Converts electrical power to angular motion.

(Noah Robertson/Melissa Haver)

Table 8: Receiver Direction Finder Level 2 Functional Requirement Table

Module	Level Meter
Inputs	-Power from battery. - Signal from directional radio antenna
Outputs	Analog voltage signal to two adjacent comparators
Function	Generates voltage signal proportional to amplitude of radio signal.
Module	**Comparator**
Inputs	-Power from battery. - Voltage signal from two adjacent level meters.
Outputs	Digital Signal.

Function	Generates digital bit indicating 'on' if first antenna is receiving a higher amplitude; 'off' if not.

Table 9: Receiver Range Finder Level 2 Functional Requirement Table.

Module	**US Receiver**
Inputs	-Power from battery. -Signal from US transducer.
Outputs	0 or 5VDC
Function	Transforms pulses of 40 kHz ultrasound to binary signal representing the presence (output 5V) or absence (output 0V) of sound.
Module	**AM Demodulator(A2) —— part #QAM-RX5-433**
Inputs	-Power from battery -Signal from omnidirectional radio antenna
Outputs	Demodulates signal.
Function	Demodulates 433 MHz AM radio signal.
Module	**Microcontroller**
Inputs	-Demodulated AM radio signal -Ultrasonic pulse envelope from US receiver
Outputs	Range information (stored in internal register)
Function	Measures time of arrival difference between omnidirectional radio antenna and ultrasonic transducer to determine range to transmitter.

(Noah Robertson/Melissa Haver)

6. PART LIST

Table 10: Part List

Qty	Refdes	Part #	Description	Unit Cost	Total Cost
1	RX_A1	SEN-10121	IMU with Accelerometer and Gyroscope	39.950	39.95
1	RX_A2	QAM-RX5-433	MODULE AM RECEIVER	7.630	7.63
5	RX_AE1-AE5	ANT-433-HETH	ANTENNA 433MHZ THRU HOLE	1.120	5.60
1	RX_AE6	PU-M4-433	ANTENNA HELICAL 1/4WAVE 433MHZ	5.000	5.00
2	RX_BT1	HR-AAUF2X5	BATT PACK 12.0V AA 1500MAH NIMH	34.330	68.66
1	RX_C1	SA115E334MAR	CAP CER 0.33UF 50V 20% AXIAL	0.114	0.11
5	RX_C10,C14, C18,C22,C26	C315C680K1G5TA	CAP CER 68PF 100V 10% RADIAL	0.390	1.95
10	RX_C11,C12, C15,C16,C19, C20,C23,C24, C27,C28	VY2100K29U2JS63V7	CAP CER 10PF 440VAC 10% RADIAL	0.280	2.80
1	RX_C2	K104K10X7RF5UH5	CAP CER 0.1UF 50V 10% RADIAL	0.029	0.03
2	RX_C3,C4	K105Z20Y5VF5TH5	CAP CER 1UF 50V RADIAL	0.079	0.16
2	RX_C6,C7	K102K10X7RF5UH5	CAP CER 1000PF 50V 10% RADIAL	0.024	0.05
1	RX_C8	K104K10X7RF5UH5	CAP CER 0.1UF 50V 10% RADIAL	0.029	0.03
5	RX_C9,C13,C17,C21,C25	S201K33S3NN63L6R	CAP CER 200PF 1KV 10% RADIAL	0.500	2.50
1	RX_D1	1N4007-TP	DIODE GEN PURPOSE 1000V 1A DO41	0.016	0.02
5	RX_D2-D6	1N4007-TP	DIODE GEN PURPOSE 1000V 1A DO41	0.016	0.08
4	RX_M1-M4	PAN14EE12AA1	MOTOR BRUSHED DC 12V 12850RPM	4.620	18.48
5	RX_Q1-Q5	2N2369A	TRANS NPN 15V 200MA TO-18	1.800	9.00

2	RX_R1	CF14JT100K	RES 100K OHM 1/4W 5% CARBON FILM	0.005	0.01
5	RX_R10,R16, R22,R28,R34	RS00550K00FE73	RES 50K OHM 5W 1% WW AXIAL	1.332	6.66
1	RX_R2	RNF14FTD15K8	RES 15.8K OHM 1/4W 1% AXIAL	0.010	0.01
1	RX_R3	RNF14FTD49R9	RES 49.9 OHM 1/4W 1% AXIAL	0.010	0.01
1	RX_R4	RNF14FAD316K-1K	RES 316K OHM 1/4W 1% AXIAL	0.015	0.02
5	RX_R6,R13,R 18,R24,R30	CF14JT3K90	RES 3.9K OHM 1/4W 5% CARBON FILM	0.005	0.02
5	RX_R7,R13,R 19,R25,R31	CF14JT1K10	RES 1.1K OHM 1/4W 5% CARBON FILM	0.005	0.02
5	RX_R8,R14,R 20,R26,R32	CMF60250R00BH EB	RES 250 OHM 1W .1% AXIAL	0.220	1.10
5	RX_R9,R15,R 21,R27,R33	PAC100005009FA 1000	RES 50 OHM 1W 1% AXIAL	0.231	1.16
1	RX_SW1	100SP1T1B4M2Q E	SWITCH TOGGLE SPDT 5A 120V	2.110	2.11
1	RX_U1	PIC16F716-I/P	IC MCU 8BIT 3.5KB FLASH 18DIP	1.210	1.21
2	RX_U2,U3	TB6561NG	IC MOTOR DRIVER PAR 24SDIP	3.920	7.84
1	RX_U4	L7812CV	IC REG LDO 12V 1.5A TO220AB	0.240	0.24
1	RX_U5	MCP1702-5002E/TO	IC REG LDO 5V 0.25A TO92-3	0.580	0.58
1	RX_U5	MCP1702-5002E/TO	IC REG LDO 5V 0.25A TO92-3	0.580	0.58
1	RX_U6	MCP6231-E/P	IC OPAMP GP 300KHZ RRO 8DIP	0.380	0.38
1	RX_U7	MCP6542-E/P	IC COMP 1.6V DUAL P-P 8DIP	0.680	0.68
3	RX_U8-U12	MCP6542-E/P	IC COMP 1.6V DUAL P-P 8DIP	0.680	2.04
1	RX_Y1	MA40S4R	RCVR 40KHZ ULTRASONIC	5.000	5.00
1	TX_A1	QAM-TX2-433	MODULE AM TRANSMITTER	4.810	4.81
1	TX_AE1	PU-M4-433	ANTENNA HELICAL 1/4WAVE 433MHZ	5.000	5.00

4	TX_BT1-BT4	CR2016	BATT LITH COIN 3V 20MM	0.360	1.44
1	TX_C1	ECE-A1HKGR15	CAP ALUM 0.15UF 50V 20% RADIAL	0.190	0.19
2	TX_C2, TX_C4	VY2100K29U2JS6 3V7	CAP CER 10PF 440VAC 10% RADIAL	0.280	0.56
1	TX_C3	K102K10X7RF5U H5	CAP CER 1000PF 50V 10% RADIAL	0.024	0.02
1	TX_C5	SA115E334MAR	CAP CER 0.33UF 50V 20% AXIAL	0.114	0.11
1	TX_C6	K104K10X7RF5U H5	CAP CER 0.1UF 50V 10% RADIAL	0.029	0.03
2	TX_Q1, Q2	2N3904-AP	TRANSISTOR NPN GP 40V TO92	0.031	0.06
1	TX_R1	MFR-25FBF52-787K	RES 787K OHM 1/4W 1% AXIAL	0.100	0.10
1	TX_R2	MFR-25FBF52-97K6	RES 97.6K OHM 1/4W 1% AXIAL	0.011	0.01
1	TX_R3	SFR16S0002261F R500	RES 2.26K OHM 1/2W 1% AXIAL	0.020	0.02
2	TX_R4, TX_R6	CF14JT1K20	RES 1.2K OHM 1/4W 5% CARBON FILM	0.005	0.01
2	TX_R5, TX_R7	CF14JT2K20	RES 2.2K OHM 1/4W 5% CARBON FILM	0.005	0.01
1	TX_SW1	V70113SS05Q	SWITCH SLIDE SPST 10A 125V	3.020	3.02
1	TX_U1	NE556DR	IC OSC TIMER DUAL 100KHZ 14SOIC	0.150	0.15
1	TX_U2	L7812CV	IC REG LDO 12V 1.5A TO220AB	0.240	0.24
1	TX_VR1	CT6EP500	TRIMMER 50 OHM 0.5W TH	0.790	0.79
1	TX_Y1	MA40S4S	TRANS 40KHZ ULTRASONIC	5.500	5.50
2		BH800S	HOLDER COIN CELL 2-20MM CELLS	1.180	2.36
1		TD-138-004	DAGU 48mm Omni Wheel Set - 4 Wheels	13.250	13.25
			Total Cost		229.40

7. PROJECT SCHEDULE

Table 11: Gantt Chart.

Task Name	Duration	Day Started	Day Finished	Who Completed
1. Midterm Report				
1.1 Accepted Technical Design Description	6 Days	10/7/2014	10/13/2014	Noah, Rob, Melissa
1.2 Marketing Requirements	6 Days	10/7/2014	10/13/2014	Rob
1.2.1 Updating Requirements	1 Day	11/20/2014	11/20/2014	ALL
1.3 Engineering Specifications	6 Days	10/7/2014	10/13/2014	Rob
1.4 FR Tables	6 Days	10/7/2014	10/13/2014	Melissa
1.5 Gantt Chart	2 Days	11/25/2014	11/27/2014	Dan
1.6 Create Parts List	3 Day	11/21/2014	11/24/2014	Dan
1.7 Format Report	6 Days	10/7/2014	10/13/2014	Melissa
1.8 Mid Term Power Point Presentation	4 Days	10/7/2014	10/13/2014	ALL
1.8.1 Formatting	3 Days	10/13/2014	10/16/2014	Dan, Melissa
1.8.2 Practice Presentation	1 Day	10/16/2014	10/16/2014	ALL
1.9 Final Report	7 Days	11/24/14	12/1/2014	ALL
1.10 Final Power Point Presentation				
1.10.1 Formatting	3 Days	11/24/2014	11/27/2014	Dan, Melissa
1.10.2 Practice Presentation	1 Day	11/27/2014	11/27/2014	ALL
2. Research				
2.1 Antenna Theory	103 Days	8/25/2014	12/5/2014	Noah
2.2 Mechanical Design	33 Days	8/25/2014	9/25/2014	Rob, Melissa
2.3 Gyroscopes	33 Days	8/25/2014	9/25/2014	Dan
2.4 Accelerometers	33 Days	8/25/2014	9/25/2014	Dan
2.5 Inertial Measurement Unity	7 Days	9/30/2014	10/7/2014	Dan
2.6 RF Transmitter and Receiver Design	92 Days	9/4/2014	12/5/2014	Noah
2.7 Ultrasonic Transmitter and Receiver Design	92 Days	9/4/2014	12/5/2014	Rob
3. Simulations				
3.1 Mechanical Design				
3.1.1 Drawing Design	27 Days	8/25/2014	9/21/2014	Rob, Melissa
3.1.2 Purchasing Parts	3 Days	9/22/2014	9/25/2014	Rob, Melissa
3.1.3 Assembling a Design	1 Days	9/24/2014	9/25/2014	Rob, Melissa
3.2 Antenna Design	65 Days	10/2/2014	12/5/2014	Noah

3.2.1 Matlab Simulation of Antenna Array	10 Days	11/1/2014	11/11/2014	Noah
3.3 Pseudo C Code Compilation	20 Days	11/3/2014	11/23/2014	Noah
3.4 Ball Design	14 Days	10/20/2014	11/3/2014	Rob
3.4.1 Designing Auto CAD Layout	5 Days	10/20/2014	10/25/2014	Rob
3.4.2 3D Printing the Ball	9 Days	10/26/2014	11/3/2014	Rob
3.7 Motor Controller	5 Days	11/15/2014	11/20/2014	Melissa
4. Calculations				
4.1 Time Trials	1 Day	9/17/2014	9/18/2014	Noah
4.2 Antenna Parameters	10 Days	9/20/2014	9/30/2014	Noah
4.3 Voltage Vector Design	35 Days	11/1/2014	12/5/2014	Dan
4.3 Mechanical Analysis	12 Days	10/01/2014	10/13/2014	Rob, Melissa
5. Testing				
5.1 Antenna Range Testing	35 Days	11/1/2014	12/5/2014	Noah
5.2 Antenna Array Tests	7 Days	11/1/2014	11/7/2014	Noah
5.3 Transmitter and Receiver Troubleshooting	7 Days	11/6/2014	11/13/2014	Noah
5.4 Ultrasound Testing	7 Days	11/6/2014	12/5/2014	Dan, Noah
5.5 Code Debugging	35 Days	11/1/2014	12/5/2014	Dan
6. Implementation				
6.1 Motor Controller Design	5 Days	11/20/2014	11/25/2014	Melissa
6.2 Compiling Parts For Transmitter and Receiver	3 Days	11/20/2014	11/23/2014	Dan
6.3 Applying Stabilizer	2 Days	11/23/2014	11/25/2014	Rob
6.4 Compiling Ultrasound and RDF Design	30 Days	11/5/2014	12/5/2014	Noah

(Dan Madden)

8. DESIGN TEAM INFORMATION

Melissa Haver, Electrical Engineering, Igor Tsukerman
Robert Haver, Electrical Engineering, Igor Tsukerman
Daniel Madden, Electrical Engineering, Igor Tsukerman
Noah Robertson, Electrical Engineering, Igor Tsukerman

9. CONCLUSIONS AND RECOMMENDATIONS

Thus far the development of this project has been focused on the design and research aspect of the Autonomous Robot Sphere. The next phase of this project will entail the implementation of the design concepts outlined in this report. A large portion of this project will depend on the testing and comparing of different possible approaches to meeting the listed design requirements. As such, the next phase of this project will also require eliminating many of these possible approaches and deciding on a final method to pursue.

10. REFERENCES

Primary Idea

[1] https://www.gosphero.com/sphero-2-0/

[2] Michaud, F.; Laplante, J.-F.; Larouche, H.; Duquette, A.; Caron, S.; Letourneau, D.; Masson, P., "Autonomous spherical mobile robot for child-development studies," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* , vol.35, no.4, pp.471,480, July 2005

[3] D. Premack, "The infant's theory of self-propelled objects," Cognition, vol. 26, pp. 1–16, 1990.

[4] Tomi Ylikorpi and Jussi Suomela, Ball-Shaped Robots, *Climbing and Walking Robots: Towards New Applications*, 2007.

[5] S. Appadwedula and C. M. Keller. Direction-finding results for a vector sensor antenna on a small UAV. Presented at Sensor Array and Multichannel Processing, 2006.

[6] Chen Zhang and Y. Lu. A realistic study of the nature inspired small aperture direction finding technique. Presented at Microwave Conference, 2009.

[7] A. Rutkowski. Small passive direction finding and IFM device. Presented at Microwaves, Radar and Wireless Communications, 2004. MIKON-2004.

[8] R. L. Johnson, J. E. Hupp and W. M. Sherrill. Radio direction finding. *Encyclopedia of Electrical & Electronics Engineering 18* pp. 64-77. 1999.

[9] Michaud, F.; Laplante, J.-F.; Larouche, H.; Duquette, A.; Caron, S.; Letourneau, D.; Masson, P., "Autonomous spherical mobile robot for child-development studies," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* , vol.35, no.4, pp.471,480, July 2005 doi:10.1109/TSMCA.2005.850596

Patents:

[18] A Halme, T Schonberg and Y Wang, "Motion Control of a Spherical Mobile Robot," in *Proc. 4th International Workshop on Advanced Motion Control, Tsu, Japan, 1996*, 1996. Patent number 960103, inventors; Torsten Schönberg, Aarne Halme

[19] Anhalt, D., & Gremban, K. D. (Apr 25, 2013). In Science Applications International Corporation (Ed.), *Methods and systems for an autonomousrobotic platform* (700/248 ed.) B25J9/16. Retrieved from https://www.google.com/patents/US20130103195?dq=autonomous+combat+robot&ei=GgYmU_P4OuPY2AXknIGABg#classifications

[20] Garrett, D. G., & Isaac, T. A. (2013). In Raytheon Company (Ed.), *Safe and arm system for a robot* (89/27.12 ed.). US: F41A19/00.

[21] F. C. Alpers. Passive direction finding system. 1980. Available: https://www.google.com.lib.ezproxy.uakron.edu:2443/patents/US4236159.

11. APPENDICES

A. Code for Antenna Array

```
#!/usr/bin/ruby2.0 -w

require_relative 'kid_control'
require_relative 'drive_control'

step_size = 0.1
length_of_simulation = 12
number_of_loops = 50

for kid_behavior in %w[line_stop]#%w[chase evade line to_from to_zig zig]
  3.upto(7) do |number_of_antennas|
    path = "#{kid_behavior}#{number_of_antennas}"
    Dir.mkdir "../#{path}" unless Dir.exists? "../#{path}"

    overall_max_speed = 0
    overall_avg_speed = 0
    overall_min_range = 2
    overall_max_range = 2
    overall_avg_range = 0
    1.upto(number_of_loops) do |loop_number|
      kid = [0.0, 0.0]
      rand_start_angle = 2 * Math::PI * rand
      ball = [2 * Math.sin(rand_start_angle), 2 * Math.cos(rand_start_angle)]

      kid_history = [kid.dup]
      ball_history = [ball.dup]

      max_speed = 0
      avg_speed = 0
      min_range = 2
      max_range = 2
      avg_range = 2 * (step_size / length_of_simulation)

      ball_x_speed = ball_y_speed = 0
      for t in (0..length_of_simulation).step(step_size)
        kid = Kid.move(kid_behavior, ball, kid, t, step_size)

        angle_to_ball = -Math.atan2(kid[0] - ball[0], -(kid[1] - ball[1]))
        angle_to_ball += 2 * Math::PI if angle_to_ball < 0
        normalized_angle = (number_of_antennas * angle_to_ball / (2 * Math::PI)).floor
        bearing = (Math::PI / number_of_antennas) + normalized_angle * (2 * Math::PI /
number_of_antennas)
        range = Math.sqrt((ball[0] - kid[0])**2 + (ball[1] - kid[1])**2)

        ball_x_speed, ball_y_speed = drive_control(ball_x_speed, ball_y_speed, bearing, range, step_size)
        ball[0] += ball_x_speed * step_size
```

```
      ball[1] += ball_y_speed * step_size

      kid_history.push kid.dup
      ball_history.push ball.dup

      speed = Math.sqrt(ball_x_speed**2 + ball_y_speed**2)
      max_speed = speed.abs if speed.abs > max_speed
      avg_speed += speed.abs * (step_size / length_of_simulation)
      min_range = range if range < min_range
      max_range = range if range > max_range
      avg_range += range * (step_size / length_of_simulation)
    end

    open("../#{path}/m#{"%03d" % loop_number}.m", 'w') do |file|
      file.puts "cd '/home/noah/Dropbox/Akron/Senior Design
Project/Simulations/antenna_array/#{path}'"
      file.puts "BALL_X = #{ball_history.map {|m| m[0].round(7) }.inspect};"
      file.puts "BALL_Y = #{ball_history.map {|m| m[1].round(7) }.inspect};"
      file.puts "KID_X = #{kid_history.map {|m| m[0].round(7) }.inspect};"
      file.puts "KID_Y = #{kid_history.map {|m| m[1].round(7) }.inspect};"
      file.puts 'DISTANCES = sqrt((BALL_X - KID_X).^2 + (BALL_Y - KID_Y).^2);'
      file.puts 'subplot(1,2,1); hold on;'
      file.puts 'scatter(BALL_X, BALL_Y);'
      file.puts 'scatter(KID_X, KID_Y, \'+\');'
      file.puts 'subplot(1,2,2);'
      file.puts 'plot(DISTANCES);'
      file.puts 'grid;'
      file.puts "set(gcf,'PaperUnits','inches','PaperSize',[14,6],'PaperPosition',[0 0 14 6]);"
      file.puts "print(1, '-dpng', '#{"%03d" % loop_number}.png');"
    end

    `matlab -nodisplay -nosplash -nosplash -r "run('../#{path}/m#{"%03d" % loop_number}.m');
exit;" 2>&1`
    overall_max_speed = max_speed if max_speed > overall_max_speed
    overall_avg_speed += avg_speed / number_of_loops
    overall_min_range = min_range if min_range < overall_min_range
    overall_max_range = max_range if max_range > overall_max_range
    overall_avg_range += avg_range / number_of_loops
  end

  open("../#{path}/stats.txt", 'w') do |file|
    file.puts "overall maximum speed = #{overall_max_speed}"
    file.puts "overall average speed = #{overall_avg_speed}"
    file.puts
    file.puts "overall minimum range = #{overall_min_range}"
    file.puts "overall maximum range = #{overall_max_range}"
    file.puts "overall average range = #{overall_avg_range}"
  end
 end
end
```

B. Code for Drive Control

```
def drive_control(x_drive, y_drive, bearing, range, step_size)
  # tunable paramters
  nominal_range = 2
  max_velocity = 5
  max_delta_speed = step_size * 4.5


  velocity_perp = x_drive * Math.cos(bearing) - y_drive * Math.sin(bearing)
  velocity_par = x_drive * Math.sin(bearing) + y_drive * Math.cos(bearing)

  delta_velocity_perp = -velocity_perp * max_delta_speed

  range_offset = nominal_range - range
  desired_velocity_par = range_offset * max_velocity
  delta_velocity_par = (desired_velocity_par - velocity_par) * max_delta_speed

  delta_speed = Math.sqrt(delta_velocity_par**2 + delta_velocity_perp**2)
  delta_speed_direction = bearing + Math.atan2(delta_velocity_perp, delta_velocity_par)
  delta_speed *= max_delta_speed / delta_speed.abs if delta_speed > max_delta_speed

  delta_x_drive = delta_speed * Math.sin(delta_speed_direction)
  delta_y_drive = delta_speed * Math.cos(delta_speed_direction)

  x_drive += delta_x_drive
  x_drive = x_drive / x_drive.abs * max_velocity if x_drive.abs > max_velocity

  y_drive += delta_y_drive
  y_drive = y_drive / y_drive.abs * max_velocity if y_drive.abs > max_velocity

  return [x_drive, y_drive]
end
```

C. Code for Kid Control

```
class Kid
  @@direction = 0
  @@to_or_zig = 'zig'
  @@to_or_from = 'to'

  def self.move(behavior, ball, kid, time, step_size)
    range = Math.sqrt((ball[0] - kid[0])**2 + (ball[1] - kid[1])**2)

    case behavior
      when 'chase'
        kid[0] += (ball[0] - kid[0]) / range * 3 * step_size
        kid[1] += (ball[1] - kid[1]) / range * 3 * step_size
      when 'evade'
        kid[0] -= (ball[0] - kid[0]) / range * 3 * step_size
        kid[1] -= (ball[1] - kid[1]) / range * 3 * step_size
      when 'line'
        @@direction = 2 * Math::PI * rand if time.round(3) == 0
        kid[0] += 3 * step_size * Math.sin(@@direction)
        kid[1] += 3 * step_size * Math.cos(@@direction)
      when 'zig'
        @@direction = 2 * Math::PI * rand if (time/4).round(3) == time.floor / 4
        kid[0] += 3 * step_size * Math.sin(@@direction)
        kid[1] += 3 * step_size * Math.cos(@@direction)
      when 'to_from'
        if (time/4).round(3) == time.floor / 4
          @@to_or_from = {'to'=>'from', 'from'=>'to'}[@@to_or_from]
        else
          if @@to_or_from == 'to'
            kid[0] += (ball[0] - kid[0]) / range * 3 * step_size
            kid[1] += (ball[1] - kid[1]) / range * 3 * step_size
          else
            kid[0] -= (ball[0] - kid[0]) / range * 3 * step_size
            kid[1] -= (ball[1] - kid[1]) / range * 3 * step_size
          end
        end
      when 'to_zig'
        if (time/4).round(3) == time.floor / 4
          @@to_or_zig = {'to'=>'zig', 'zig'=>'to'}[@@to_or_zig]
          @@direction = 2 * Math::PI * rand
        else
          if @@to_or_zig == 'to'
            kid[0] += (ball[0] - kid[0]) / range * 3 * step_size
            kid[1] += (ball[1] - kid[1]) / range * 3 * step_size
          else
            kid[0] += 3 * step_size * Math.sin(@@direction)
            kid[1] += 3 * step_size * Math.cos(@@direction)
          end
        end
      when 'line_stop'
```

```
      @@direction = 2 * Math::PI * rand if time.round(3) == 0
      @@stop = true if (time/8).round(3) == time.floor / 8
      @@stop = false if time.round(3) == 0
      if !@@stop
        kid[0] += 3 * step_size * Math.sin(@@direction)
        kid[1] += 3 * step_size * Math.cos(@@direction)
      end
    end

    return kid.dup
  end
end
```