Ballbot Fall Semester Design Report:
Design Process

Graham Goodwin, Adam Woo, Chloe Desjardins

Capstone I: ENGR 483

Professor Huang and Professor Mertens

December 15th, 2017

Table of Contents

I. History of the Ballbot

In 2008, a team of graduate researchers at Carnegie Mellon University developed the first omnidirectional robot, capable of interacting in human environments. This design differed from other robotic designs in that for a robot to actively interact with human movement, an entirely different approach to locomotion must be implemented. The goal was to develop mobile robots that are safe, dynamically agile, and capable of graceful motion. The robot would be slender enough to easily maneuver in crowded, peopled environments, and readily yield when pushed around. In order to meet these design requirements, the team of researchers decided to create an intrinsically unstable robot which uses a ball for mobility. This new concept introduced the family of robotics to "ballbots." Despite the capabilities of the omnidirectional machines known as ballbots, there has been little improvement made on the original designs.

In addition to the work done by the researchers at Carnegie Mellon University, some engineering students at ETH in Zurich have created a similar ballbot, using the concepts of stability and balance. This ballbot, named "Rezero," has the ability to interact with humans, aims at high agility and fast movement, as well as create a hands-on experience with the Ballbot technology. Rezero's dynamic hull allows the ballbot to display emotions such as breathing, being frightened, or waking up and going back to sleep.

As the ballbot technology continues to improve, there are a variety of ways that this type of robot can be advantageous in everyday life. For example, ball bots can serve as guides in museums or act as security monitors in public places. They can also be used as a daily aid, performing basic tasks such as transporting and delivering objects to humans. Ballbots can also be used as service robots assisting the disabled such as the blind. Due to the ballbot's ability to navigate in cluttered environments and perform interference correction, the ballbot could potentially assist patients in hospitals.

II. Problem Definition

The primary objective of this project is to develop a single-contact, self balancing robot that is capable of omnidirectional movement.

III. Problem Evaluation

In order for this robot to perform in practical applications, several design criteria must be met. The ballbot must be slender enough to enable it to maneuver through crowded spaces. The body of the robot cannot be too wide, as the robot should take up as much walking space as a human at most. The ballbot must also be tall enough to interact with a standing human. Lastly, the ballbot needs to be capable of performing interference correction. This means that the ballbot can give in to small pushes, but then

reset and continue on its trajectory. These three design goals correlate with the potential use cases for ballbots in that our ballbot will be capable of omnidirectional stabilization and agile movement in crowded environments.

We decided early on that humans should be able to easily interact with our system while standing, thus one of our primary design goals was to make the system at least three feet in height. Additionally, there would be little need for a single contact system unless it holds some advantages over multiple contact systems. A primary advantage of single contact systems is that the base can be quite narrow, even if the entire body is rather tall. This means that such systems can easily maneuver through crowded environments without sacrificing height. To play to this strength, we decided to aim for a system that would be slender, less than one foot wide.

Some goals were discussed but not established, such as speed, precision, carrying capacity, durability, interactability, and power consumption. However, we decided that these extra design goals were unnecessary or impractical for our purposes. These specifications were set as considerations, with the intention to optimize them as much as possible, but not to establish a specific requirement for project completion. With the design goals established, we moved on to address the design constraints of this project.

The first constraint established was time. The time limit for this was defined as slightly less than an academic year, accounting for presentations and analysis and the end of the year. Our group agreed that this was ample time for the projects main objective of designing a stationary balancing robot, with room to add movement or other aspects later down the line.

The second constraint addressed was cost. The Trinity College Engineering Department allocated $150 per student for their senior design projects. That means that our team of three was given a $450 budget. A quick look at the required equipment we might need to buy allowed us to estimate that we would have more than enough money to complete this project.

IV. Alternatives

Before weighing alternative parts for our system, we first had to decide on an overall design of the system. We looked at several examples of existing ballbots, some of which are displayed below:

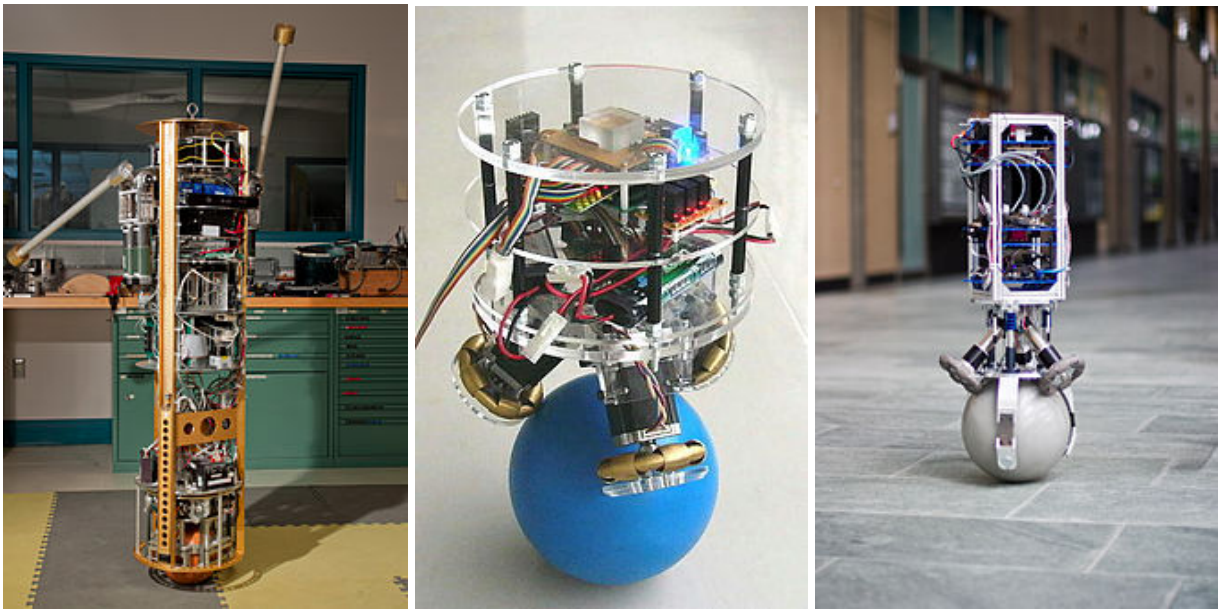


Figure 1: Carnegie Mellon University Ballbot (left), Tohoku Gakuin University Ballbot (center), The Rezero (Right)

These robots uses a combination of sensors to measure the position of the robot and calculate the necessary motion to keep them stable. After analysing several systems and comparing different arrangements, we decided to break down the system like this:

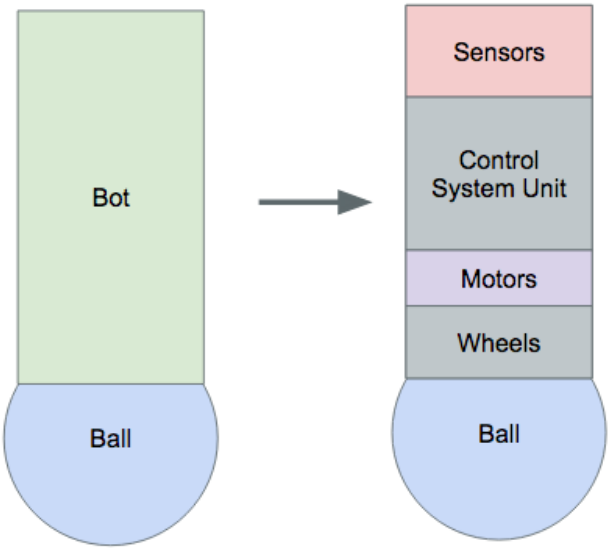


Figure 2: Initial ballbot system breakdown

The motors and battery should be placed as low as possible on the robot as to keep the center of gravity closer to the axis of rotation, and the sensors should be at the highest point in order to experience more

exaggerated movement for measurement. The components of the control system should weigh little enough that their placement should have a negligible impact on the system's stability.

With a tentative structure of the robot designed, we began analysing alternatives for each component.

The Ball

From our design goals, we knew we wanted the base of the system to be less than one food wide, and thus we only considered types of balls whose diameters were about that wide.

From our problem evaluation, we knew that a lightweight ball would require less torque from the motors to move. Thus, we tried to strike a balance between weight and functionality when choosing the ball. We researched the following alternatives, and listed the pros and cons of each:

Table 1: Ball Alternatives with Pros and Cons

Name	**Pros**	**Cons**
Bowling Ball	-rigid surface -good quality control (close to perfectly spherical)	-slippery surface -requires excessive torque from motors -expensive (~$40)
Steel "Mirror Ball" Lawn Ornament	-hollow (less torque required) -rigid surface	-slippery surface -expensive (~$30)
Hollow Plastic Ball	-low mass (little torque required for motion) -Inexpensive (<$10)	-flexible surface -slippery surface
Basketball	Inexpensive (~$10) -low mass -surface has high coefficient of friction	-ridges on surface introduce error/potential failure of system -not entirely spherical

While each of the alternatives above have their own merits, we decided that we could best work around the shortcomings of the steel mirror ball. The bowling ball had all of the shortcomings of the mirror ball in addition to the excessive torque requirements it would place on the motors. The benefits did

not seem to outweigh this issue, so we eliminated the bowling ball alternative. A bowling ball is also relatively expensive ($40), thus we decided that this was not the best fit for our system.

The only real benefit of the hollow plastic ball was that it was extremely cheap, however, since the other alternatives were well within our budget, price was not a major concern. So we quickly eliminated the plastic ball alternative as well.

The basketball was a close contender with the mirror ball. It would provide good grip for the wheels on the surface of the ball, and was lighter than the mirror ball by about 0.3 kg. However, the ridges on the basketball would introduce error into our system. It would be near impossible to predict when the wheels would be rolling over one of the basketball ridges, and thus some varying degree of unknown error would be introduced. We were already getting a sense of the complexity of the mathematical system behind the ballbot, so we decided to pursue the mirror ball alternative. To counteract the slippery surface of the mirror ball, we came up with the idea of coating it in plastic spray. This would eliminate the surface grip problem, and since the price was within our original cost estimate, we were satisfied with this alternative.

Chosen Alternative: Steel mirror ball with plasti-dip coated surface

- Size: 10” diameter (near perfect sphere)
- Weight: 0.91 kg
- Total Cost: $45 ($35 + $10 for plasti-dip)

The Microcontroller

In order to control a dynamically stable system, the microprocessor needs to be able to communicate with sensors, calculate the raw data through a control system, and send the control signals to the motors extremely quickly. The main issue with this design is that the control system is complex, involving several trigonometric calculations and filters which are very processor intensive. The microcontroller needs to have a processor capable of computing these calculations quickly.

Additionally, we wanted to program the controller using our language of choice, C++. We use the Atom text editor plugin PlatformIO to write, build, and upload our programs to various microcontrollers. While this is not a restriction for our system, it was an ease of use factor which influenced our decision.

To decide on a microcontroller, we researched what had been used for several other ballbots. The main alternatives to decide between were as follows:

Table 2: Microcontroller Alternatives with Pros and Cons

Name	**Pros**	**Cons**
Arduino Mega	● already owned (~$10 otherwise) ● PlatformIO bootloader compatible ● 54 programmable pins	● 8 Mhz clock speed
Raspberry Pi 3 Model B	● 1.2 GHz clock speed ● 26 programmable pins ● Large community ● USB power ● Expandable	● expensive (~$35) ● Complex library setup ● Not PlatformIO compatible
Teensy 3.6	● 180 MHz clock speed ● PlatformIO bootloader compatible ● 32 programmable pins ● Arduino library compatible ● Micro USB power port	● expensive (~$40)

After extensive research on what microcontroller would be the most efficient to use, we decided to choose the Teensy 3.6. Initially, we were going to use a Raspberry Pi because of its familiarity and its extensive community support. However, after performing further problem evaluation, we decided to go with the Teensy 3.6 due to its larger number of programmable pins (32), high processing speed (180 MHz), and good compatibility with our IMU and it's fusion libraries (more on this later).

Chosen Alternative: Teensy 3.6

- Processor Speed: 180 MHz
- Cost: $40

The Ball Drive System

The parts in contact with the ball needs to allow movement in all directions. The parts cannot restrict range of motion in any why. The type of ball drive system was relatively easy to chose, as majority of the alternatives were either too difficult to build, or would not work all together.

Table 3: Wheel Alternatives with Pros and Cons

Name	**Pros**	**Cons**
Partially Sliding Rollers	● Inexpensive	● Cannot be purchased ● Difficult to build
Omni-wheels	● Easily purchased ● Passive lateral movement	● Low strength ratings for cheaper models ● Non standard mounting options
Mecanum wheels	● Easily purchased ● Inexpensive ● Higher strength ratings	● No passive lateral movement (active movement only)
Spherical Induction Motor	● Single moving part ● Highly accurate motion	● Cannot be bought ● Very complex to build

Omni wheels were the only movement system that was feasible for this project. The mecanum wheels restricted movement laterally, and the other two systems were two difficult to design in the given time period. While the type of ball drive system itself was easy to chose, the design of this system was more difficult to agree upon. We looked at several different orientations for the wheels:

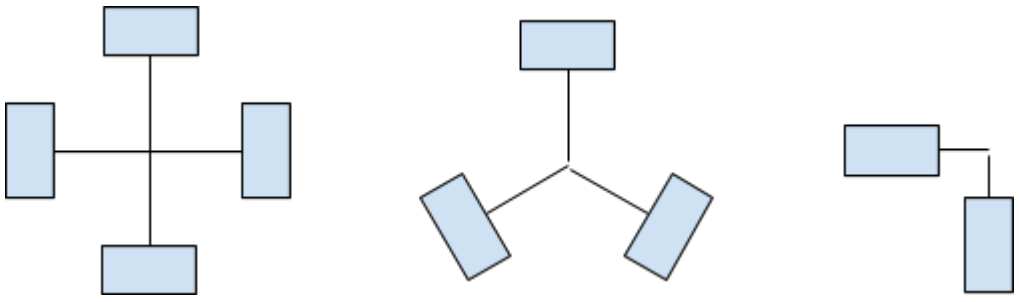


Figure 3:Various wheel arrangements

The first orientation we looked at was the four wheeled design which maps two wheels to each axis. This was a good design and easy to control. However, omni wheels and the motors required to drive them are expensive, as well as heavy. To cut down on these costs, we looked at a two wheeled model which used point the wheels towards the center of the robot. This gave us control over the x and y axis, but removed the possibility of rotation. The compromise was a three wheeled design spaced evenly apart, which the wheels placed perpendicularly to the robot's center. The wheels would be angled at 45 degrees and placed 45 degrees from the ball's center to maximize control over all axis of rotation.

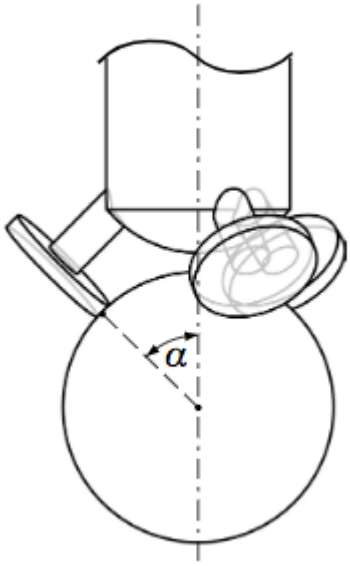


Figure 4: The three wheeled design of the ball drive system, where α=45°

This design maximizes control over the ball, while minimizing the weight and cost of the system.

Chosen Alternative: Three omni wheels

- Manufacturer: Vex Pro
- Size: 3.25" diameter
- Total Weight: 0.54 kg (3 wheels, 0.18 kg each)
- Total Cost: $54 (3 wheels @ $18 each)

The Angle Measurements Devices

For these type of projects, inertial measurement units (IMUs) are the most widely used by far. There really were no other alternatives to decide between for measuring the robot angle from vertical. Among IMUs, the MPU6050 and MPU9250 were the two alternatives we considered.

Table 4: IMU Alternatives with Pros and Cons

Name	**Pros**	**Cons**
MPU6050	● 6 axis (accelerometer, gyroscope) ● Inexpensive ($7)	● -no magnetometer (cannot measure compass direction)
MPU9250	● 9 axis (accelerometer, gyroscope, magnetometer) ● More precise measurement after filter	● More complex filters ● Less community support ● Less board compatibility ● Slightly more expensive ($10)

The MPU9250 was the most suitable option compared to the MPU6050. Although the MPU6050 is three dollars cheaper than the MPU9250, the MPU9520 has 9 axes with an accelerometer, gyroscope and magnetometer. The MPU9520 is also capable of measuring compass direction. When considering our future goals for the movement of our ballbot, these additional capabilities of the MPU9520 were enticing and seemed more practical for our ballbot design. The price of the MPU9520 was also still within our economic means.

Chosen Alternative: MPU9250

- Functionality: 3-axis accelerometer (angular acceleration), 3-axis gyroscope (angular velocity), 3-axis magnetometer (compass direction)
- Cost: $10

The Motors

To drive the wheels, we required motors with relatively high torque at low speeds that could handle precise movement and change direction quickly. This indicated to us that stepper motors would likely be the best choice, though we did consider brushless DC motors as well.

Table 5: Motor Alternatives with Pros and Cons

Name	**Pros**	**Cons**
DC Brushless Motors	• Inexpensive • High RPM • PWM	• Lower torque • Ramping acceleration
Stepper Motors	• High torque at low speeds • Precision movements	• More expensive • No PWM

We decided to use stepper motors for our system, because of their high torque at low speeds and precise movement options. We decided to use an a4988 stepper motor driver to control and power them. We also decided to use a motor that fits the NEMA 23 standard. This allows us to start designing the mounting hardware, even though we do not know the exact model we will be using.

We have not yet decided on a model because we want to make sure the rated torque is high enough for our system, which requires the math modeling to be completed first. The NEMA 23 comes in a variety of lengths, the longer the motor the higher the torque. We will purchase the motors early next semester.

Ballbot Fall Semester Design Report:
Full System Design Description

Graham Goodwin, Adam Woo, Chloe Desjardins

ENGR 483: Engineering Capstone I

December 15, 2017

Introduction

The following document details our current full-system design. It serves as an overview of the subsystems we have identified, as well as the completion levels of those systems.

Current System Design

We identified the following subsystems of the ballbot:

1. Chassis
2. Inertial Measurement Unit System (9-axis IMU)
3. State observer system (microprocessor)
4. Controller system (microprocessor)
5. Ball drive system
6. Power system

After going through the problem evaluation process and weighing alternatives, we arrived at our current full-system design. A physical layout of the system can be seen below.

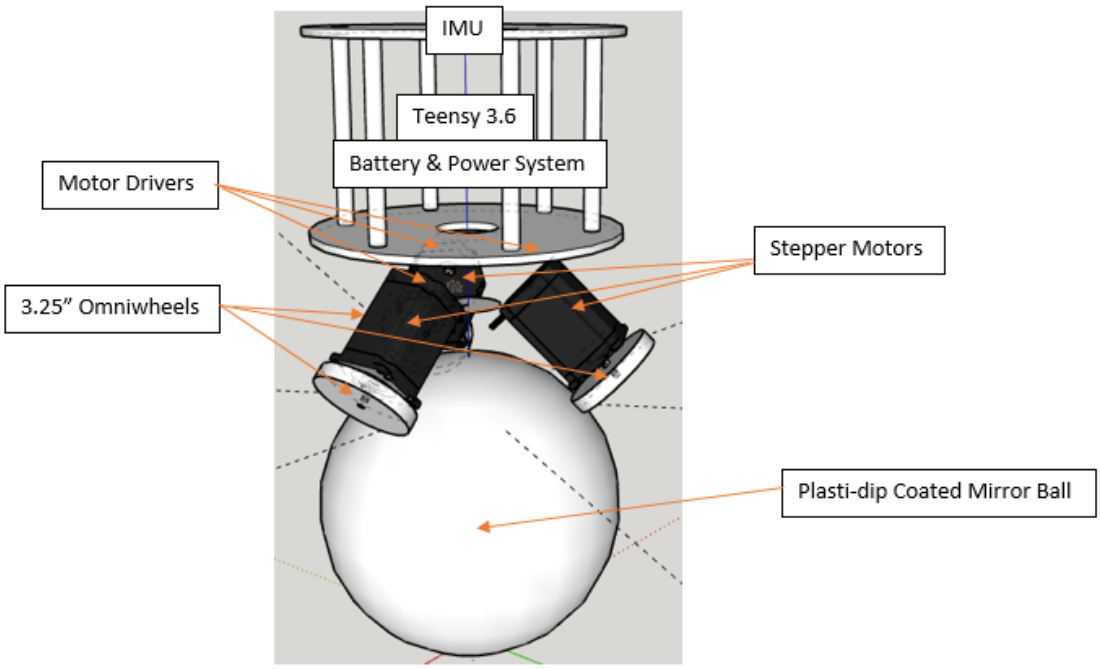


Figure 5: Full-system Physical Layout

We also constructed a block diagram depicting how the system components will interact (seen below).

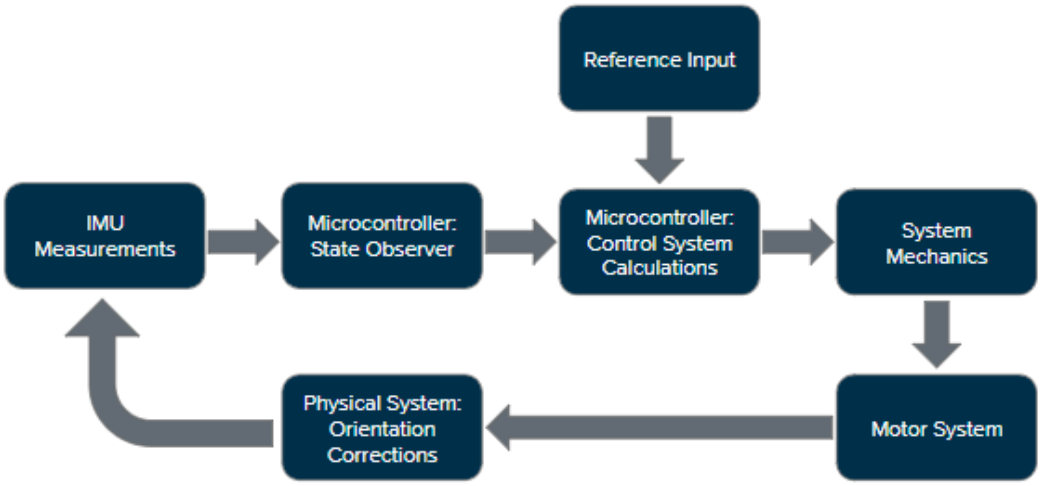


Figure 6: Full system block diagram

Chassis

The chassis design for the system was kept very simple with only the purpose of housing the other components. We chose a 10” diameter circular platform design that can be easily raised and lowered depending on the size of the parts. One platform will attach to the motor mounts, and hold the battery and motor drivers on it. The middle platform will hold the microprocessor, and the top platform will have the IMU mounted to the bottom of it. The platforms will be held together using six ¼” threaded rod and held in place with twelve ¼” nuts and washers for each platform. This allows the platforms to be moved according to the needs of the system, but still remain secure when desired.

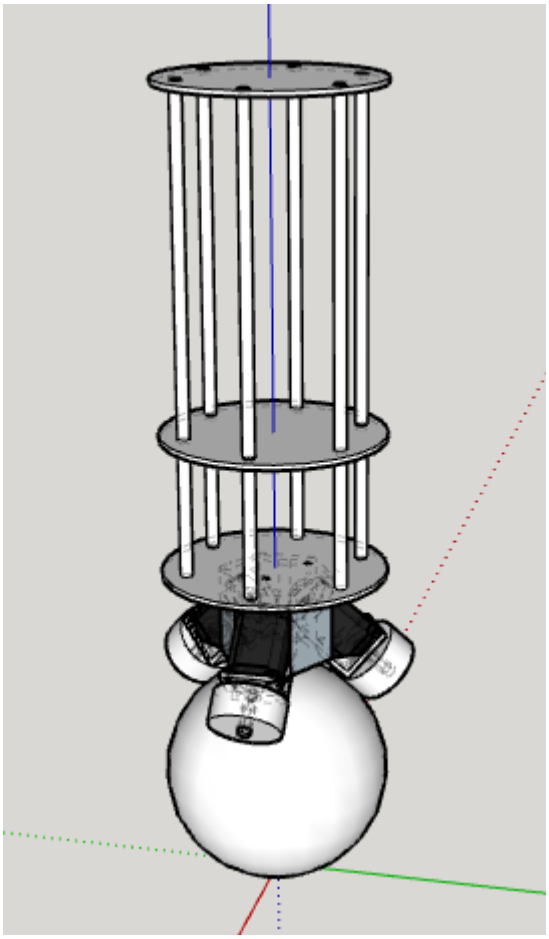


Figure 7: Full size CAD model of the ballbot

IMU System

The MPU9250 consists of an accelerometer, gyroscope, and magnetometer, which read angular acceleration, velocity, and compass readings respectively and communicates them to the microcontroller using I2C. These sensors supply three raw data points each, one for each axis, for a total of nine data points. In order to convert these data points into understandable Euler angles (pitch, roll, and yaw), we need to first put these points through sensor fusion, a method of combining the readings to reduce noise and minimize calculation error. Our chosen method of sensor fusion was Sebastian Madgwick's Orientation Filter, seen below:

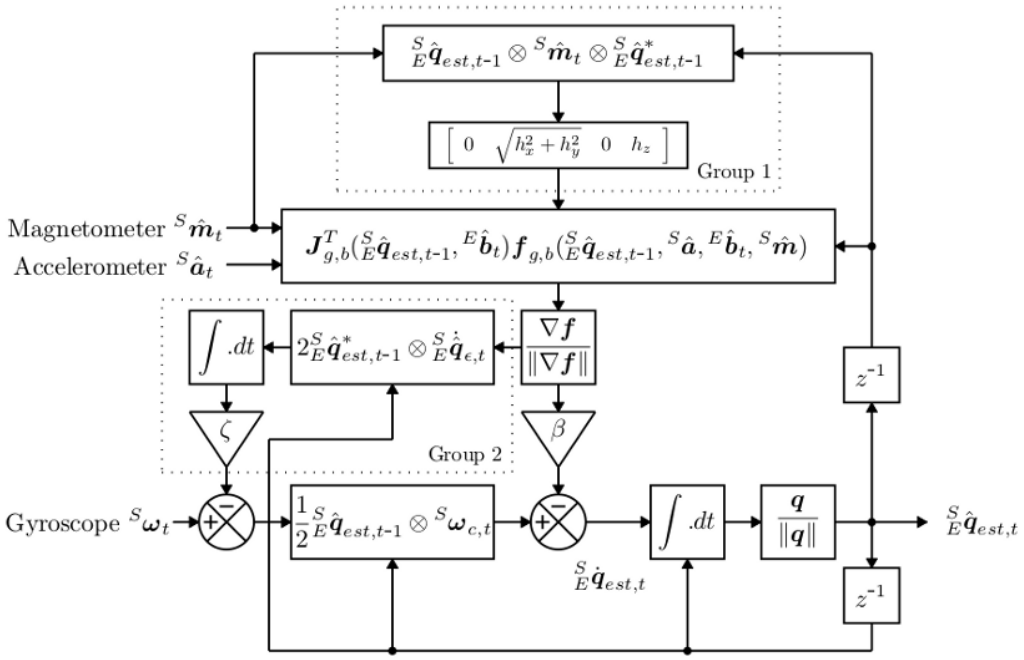


Figure 8: Sebastian Madgwick's Orientation Filter

This sensor fusion algorithm inputs the data from the three measurement tools and outputs a quaternion, which is essentially a four dimensional unit vector. This quaternion can be used to find Pitch, Roll, and Yaw using the following equations:

$$Roll = \phi = \arctan\left(\frac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)}\right)$$
$$Pitch = \theta = \arcsin\left(2(q_0q_2 - q_3q_1)\right)$$
$$Yaw = \psi = \operatorname{atan}\left(\frac{2(q_0q_3 + q_1 + q_2)}{1 - 2(q_2^2 + q_3^2)}\right)$$

This method was chosen because it is one of the most accurate methods for calculating the angular position. It is also open source which allowed us to use pre existing C++ libraries and modify them for our specific IMU. We have already implemented this our our microprocessor and tested the accuracy of our readings. Sebastian Madgwick claims that this algorithm is precise to less than a degree of error.

State Observer

The state observer system is a project for next semester, but we know that it will take in the Euler angles generated by the IMU calculations and output the system states, i.e. the angle from vertical and its velocity, and the angular position of the ball and it's velocity.

Control System

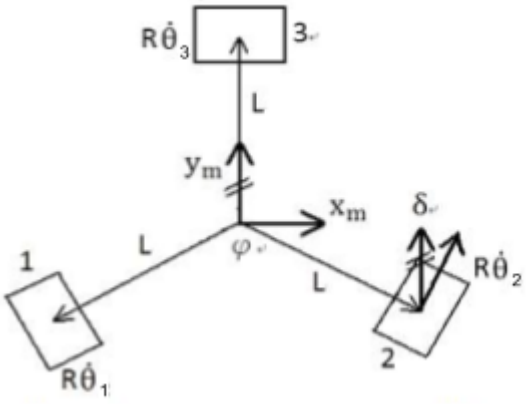
We have explored several options for controlling our system, but have yet to decide on a final method. The first method would be to use a proportional integral derivative control (PID) to control by calculating the required gains for our unique transfer function, and then refining the controls empirically. We also explored linear quadratic regulator (LQR) controls, which is an optimizing form of control that aims to minimize some user-defined cost function. In our case, that cost function would be the systems angle from vertical.

Regardless of the control method, the inputs would be the desired angles from vertical in the z-x and z-y planes. The feedback would come from the IMU which depending on the difference between the desired angle and the actual angles, return the required x and y velocities for the ball to stabilize the system. The ball drive system will take the desired x and y velocities and convert them to the appropriate motor speeds to move the ball the desired resultant vector.

Ball Drive System

The ball drive system is made up of the motor drivers, the motors, and the wheels. The a4988 stepper motor drivers power the motors and control their speed via their connection to the microprocessor. Stepper motors have a high torque at low speeds which will allow for quick changes of direction. The wheels are connected the motors by a custom designed 3d printed wheel hub.

We opted for a three wheeled design to cut down on weight and cost, as opposed to a four wheeled design. Despite the mechanical advantages of this orientation this design does presents a difficulty in controlling the motors there are no two wheels mapped to each axis. This means that a series of motor speed equations will need to be developed to move the ball correctly. By drawing out the wheel orientations as such:



$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} -\sin(\delta+\phi) & \cos(\delta+\phi) & L \\ -\sin(\delta-\phi) & -\cos(\delta-\phi) & L \\ \cos(\phi) & \sin(\phi) & L \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\phi} \end{bmatrix}$$

Figure 9. Kinematics of robot movement in the x-y plane

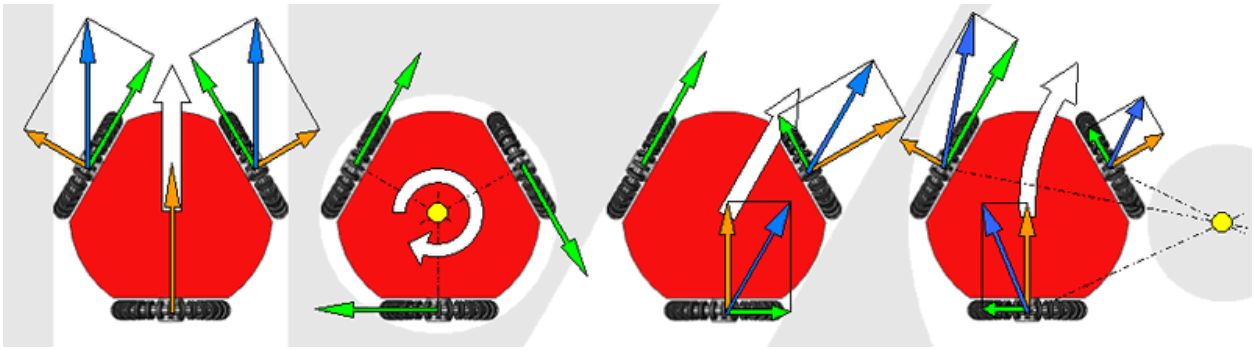
Here, δ is the angle between the wheel and the axle (90°), and ϕ is the angle between the center of each wheel (120°). This leaves us with these equations:

$$\dot{\theta}_1 = \frac{\frac{1}{2}\dot{x} - \frac{\sqrt{3}}{2}\dot{y} + L\dot{\phi}}{R}$$

$$\dot{\theta}_2 = \frac{\frac{1}{2}\dot{x} + \frac{\sqrt{3}}{2}\dot{y} + L\dot{\phi}}{R}$$

$$\dot{\theta}_3 = \frac{\dot{x} + L\dot{\phi}}{R}$$

Where $\dot{\theta}$ is the angular velocity of the motor, L is the distance from the center of the robot, and R is the wheel radius. We can find trigonometric equations that sum the velocity vectors of the motors to give a single velocity vector in a desired direction from the robot's center:



http://3.bp.blogspot.com/_a_6DwTj3ij0/TRxY3TovHHI/AAAAAAAAAGc/OVVlBWgrc5g/s1600/800px-robot_omnidirectional_movement_speedvectors.gif

Figure 10: Examples of potential movement patterns for the ball bot as a sum of the wheel velocities (green arrows)

Because we know the size of our wheels and our ball, we can find the values of L and R to plug into these equations:

$$L = \sqrt{\frac{\sqrt{10in^2 - (5in(sin(45)))^2}}{\pi}} \approx 1.755in$$

$$R = \frac{3.25in}{2} = 1.625in$$

Which leaves us with:

$$\dot{\theta}_1 = \frac{\frac{1}{2}\dot{x} - \frac{\sqrt{3}}{2}\dot{y} + 1.755\dot{\phi}}{1.625} \approx 0.308\dot{x} - 0.533\dot{y} + 1.08\dot{\phi}$$

$$\dot{\theta}_2 = \frac{\frac{1}{2}\dot{x} + \frac{\sqrt{3}}{2}\dot{y} + 1.755\dot{\phi}}{1.625} \approx 0.308\dot{x} + 0.533\dot{y} + 1.08\dot{\phi}$$

$$\dot{\theta}_3 = \frac{\dot{x} + 1.755\dot{\phi}}{1.625} \approx \frac{\dot{x}}{1.625} + 1.08\dot{\phi}$$

These equations can be used to take the desired system velocities given by the controller, and convert them into equivalent motor speeds.

The omni wheels have a rubber texture on the rollers, and the ball is coated in plastidip to increase its traction.

Power System

The power system, so far has been an afterthought. We know that the motors operate on 5.4V at 1.5A, while the Teensy 3.6 runs on a 5V input. We need to do more research on the battery and power supply system, but the basic wiring for the motors and drivers will look like this:

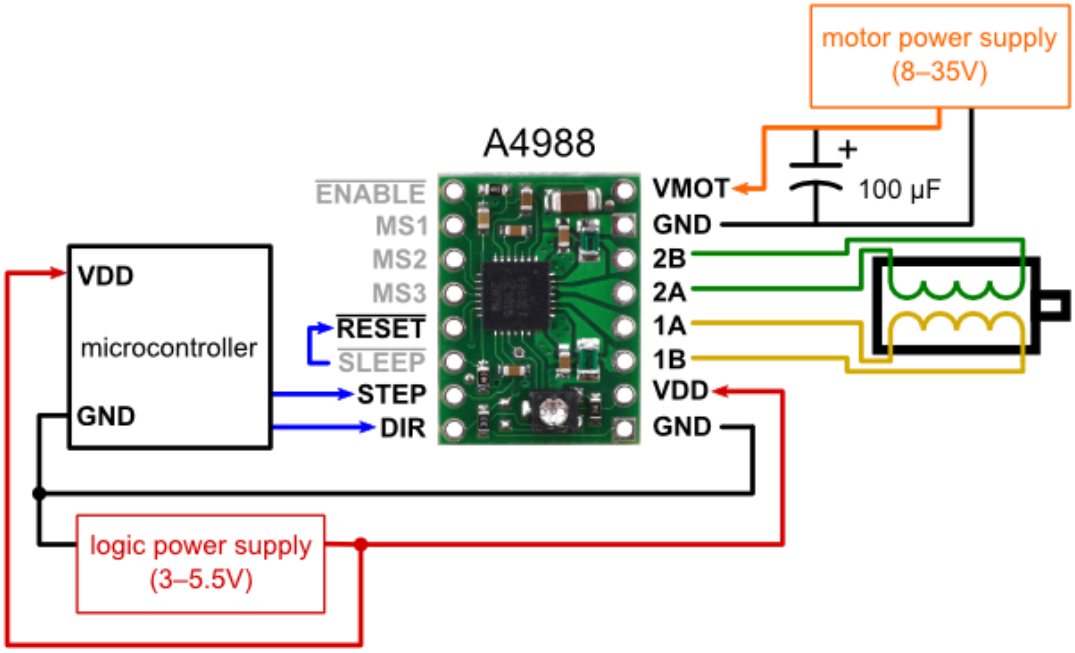


Figure 11: Circuit diagram for the a4988 motor drivers

Math Modeling

We started with a 2D planar model of the system. We start with a 2D model because the xz and yz planes of the robot are identical. The full system is then controlled by breaking each motor force into components in the xz and yz planes.

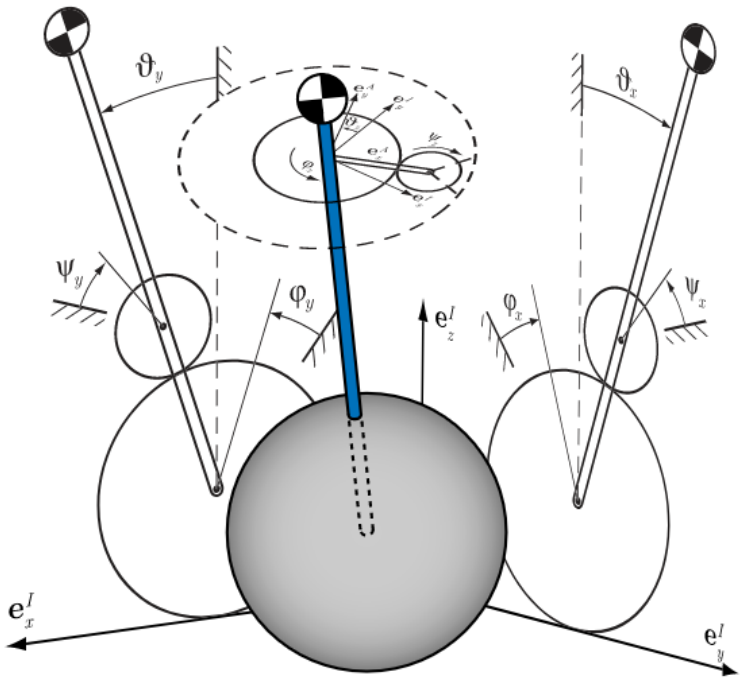


Figure 12: The 2D breakdown of the 3D system

Analyzing the system in this manner will give us a desired velocity for the x and y components and sent direction to the motor equations. The predicted result will be 3D stabilization as the combination its components.

We have not yet explored the intricacies of coupling these planar models, but the hope is that this method will give a close enough estimation of the system controls to be modified empirically.

System Completion Percents

To assess our work over the past semester, as well as clarify what needs to be done going forward, we assigned completion levels to each of the subsystems we identified. For each subsystem, a percentage has been assigned to signify how complete it is, and a second percentage has been assigned to show if the physical components have been acquired yet. These can be seen below in Table 1, along with some details on what is complete, and what is left to do.

Table 6: Subsystem Physical and Design Completion Levels

Name of System	Level of Completion	Parts Acquired	Details
Chassis	80%	0%	CAD model finished, parts are not 3D printed
IMU System	100%	100%	IMU purchased, code implemented to output pitch, roll, yaw
State Observer	10%	N/A (microprocessor)	States decided upon, need to research how to build state observer
Controller System	50%	N/A (microprocessor)	PID and LQR researched, need to incorporate it into our model
Ball Drive System	80%	60%	Ball, plasti-dip, omniwheels and motor drivers purchased, need to purchase motors
Power System	10%	0%	Need to decide on and purchase battery
Math Modeling	60%	N/A	2D model completed, need to research coupling between 2D planar models

Spring Semester Goals

After assessing our progress thus far, we have a variety of goals that must be considered and achieved for the spring semester. Firstly, we need to complete the modeling of the system such as the Computer Assisted Design (CAD) modeling and the design of the chassis. After performing various calculations on the motors this semester, we have three separate motor equations, one for each motor. These motors that will be used to move the omni-wheels must still be purchased. Although we have already acquired majority of the hardware for our system, we still need to obtain several components such as the battery. Once all necessary parts are secured, we can then assemble the physical system. We will also complete the controller design and implement the control in code using C/C++. Once all of these goals are achieved, we will then proceed to test the system and tune the controller in order to achieve stability. Lastly, if time permits, we will ultimately implement remote controls or tracking for our ballbot.

Appendix

Meeting Minutes

Below is a summary of our meeting minutes with Professor Huang. We met (officially) every Friday at 10 am in order to discuss our progress and set our goals for the following week.

Capstone Meeting: Friday 11/03/17 - all members present
Plaster dip or rubber spray for the ball - cheapest
Stepper motors - need enough torque to get ball moving
Need stepper motor controllers
Need inputs
Stepper motor driver - power driver individually, need 4 pins, can do it directly from power source
Need some sort of wedge to provide for angles, mount it to bottom of platform - use aluminum

Ball - $35
-10 in diameter

Omni-wheels -
-2-4 in in diameter?

How are we going to mount omni-wheels to the motors?
-Axle mount?

Arduino coded in C++ so we can convert to Raspberry Pi if needed
-Arduino is cheaper than Raspberry Pi
-Make sure it works with Arduino first because we are all familiar with it

Want a prototype by the end of the semester!!!

Batteries
RC batteries

Acrylic components, 3D printing a prototype test base

Equations
Kinematic equations for motors, not sure if we need them to actually make the robot work, throw PID onto board and it will compensate
-Need to model how our control inputs are going to affect our states
Model using physical equations- how the motors will affect the ball moving around, torques and pendulums
Control theory- look into building an observer, get into algebra and talk about observability and controllability space

Look over theses for designing a ball bot to get an idea of what equations of motions will look like

Things we need to accomplish for next week:
-Look into ACS stuff
-Modeling/Kinematic Equations
-Design chassis (base)
    -Axle mounts
-Ordering parts: ball, new IMUs, motors, driver chip A4888
-Find suitable batteries

Assignments:
Graham
-Modeling/Kinematic Equations

Adam
-Design chassis (axle mount)
-Begin coding
-Kinematics Equations for motor speed

Chloe
-Look into ACS stuff
-Find suitable batteries

Mutual:
Ordering ball, IMUs, motors, driver A4888, rubber coat
-Order ball, IMUs, driver A4888, rubber coat ($5) hold off on omni-wheels

Meeting minutes Friday 11/10/17 - all members present

- Model 3 motors on the ball separately using trigonometry in 3D then sum them together

-Forces from the wheels
-Pendulum falling in the plane
-Then put these equations in block diagram as parameters
-3 planar model method

- Write state space equations, block diagram, feedback of 1

-Design for particular step response
-8 states total

- Response time - assume instantaneous
- Estimator? LQR/LQG?
- PID Controller

For the next 2 weeks:
-Motors
-Wheels
-Consider use cases
-Continue math modeling, research coding techniques

Graham:
-Continue math modeling

Adam:
-Researching coding techniques
-CAD model chassis
-CAD model and print custom mounting hardware

Chloe:
-Look into use cases
-Batteries

Mutual:
Purchase motors and omni-wheels

Meeting Minutes Friday November 17 - all members present

- Raspberry pi came, MPU
- More math modeling - turn into 3D
- Inverted pendulum project?
- NiMH 3-5 Ah - look up what hobbyists have used
- Lithium iron phosphate - any 5 Ah
- Buy batteries cheap to buy a few and keep cycling them
- Mechanical model by the end of the semester in google sketch-up
- Bought: raspberry pi, plasti-dip, ball, smaller electronics
- Stepper motors, have drivers
- Omni-wheels and motors are most important
- Heavier or lighter wheel based on inertia?
- Omni-wheels on the smaller side to get a faster response, need faster RPM : 2-4 in in diameter

Progress presentation for next Tuesday (post Thanksgiving) - start working on this
Continue with what we are doing….

Things we need to do:
-Buy omni-wheels, motors, batteries
    -hobbyking.com, look at best selling battery and buy it
-CAD modeling, MPU testing
-Start using Raspberry pi

Meeting Minutes 12/4/17 - Professor Mertens, all members present

- Need to calculate max angular acceleration that we want to impart on the ball
- Find torque in relation to axis of the ball
- The torque that the motor needs depends on the motor arm
- Ball moves by applying a tangential force
- α = ??, how quickly must we accelerate the ball?
- **Need to account for slippage**
- Quantify our design criteria - create a robust theoretical model
- “Subprojects” - a) dynamic analysis of the moving ball, b) microprocessor, c) motors, d) feedback loop
- For Traveler’s Presentation on 12/12/17: Show current design, components, here is what we have completed…
    - Make sure to highlight which topic from the outline we are discussing during the presentation...Makes it easier for the listener to follow along and makes things less complicated