University of Central Florida
Department of Electrical and Computer Engineering

KittyBot

Stephen Barth - EE
Bryen Buie - CpE
Carlos Garzon - CpE
Trenton Williams - EE

Table of Contents

1 Executive Summary

The purpose of this document is to detail the design and development of the project by Group 5 of Senior Design 1 (EEL 4914) undergraduate course at the University of Central Florida (UCF) during the Summer 2016 semester. For the course, Electrical and Computer Engineering students of the University of Central must join together in teams to conceptualize, design, and finally, build a system or device that displays the engineering knowledge and skills we have gained. This document will also discuss the research that went into the project, review the applicable standards and constraints, detail the hardware and software design decisions, as well as detail the prototyping process.

Group 5 has decided to work on a project that would not only challenge us, but also be useful and fun. The project will be a robotic device whose primary function is to interact and play with cats. Cats are often times curious and playful creatures. Their interactions with the robot would be entertaining for both the cats and their owners. The robot, affectionately dubbed “Kitty-Bot”, will function as an advanced robotic toy for cats. It will be able to autonomously roam about an indoor space. It will also be able to sense its surrounds so it will not run into people, pets, or objects like walls, tables, couches, etc.

Since the primary target for the robot is cats, it will be designed with this animal in mind. It will be of a small enough size (no more than 10 inches in height) as to be an appropriately-sized plaything for the average household cat. The robot will need to be durable enough to withstand rough contact from the animal. Cats have sharp claws and teeth, so the outer shell of Kitty-Bot must be scratch resistant, and the sensitive components such as microcontrollers, printed circuit boards (PCB), and wiring will need to be housed in durable compartments. Kitty-Bot may potentially be turned over while a cat is playing with it. If this happens, Kitty-Bot will be able to set itself upright again. This will be achieved by Kitty-Bot's spherical design. It will, in essence, be a "smart ball", an autonomous, self-rolling sphere.

The project's nature of being a device for pets means that it will interact with living beings. Because of this safety is of the upmost importance. Group 5 will design Kitty-Bot to not cause physical harm to pets or their owners. It will be an indoor device as well, so very high-power will be avoided to lessen the chances of physical harm and damage to property.

Throughout this document we will explore the potential capabilities of KittyBot.

2 Project Description

2.1 Project Motivation and Goals

Every six in 10 Americans is a pet owner. Census done by the Humane Society of the United States shows that there are 86.4 million cats in households around the United States. That alone is a large population, but combined with the hundreds of millions of cats living in households across the global the numbers become staggering. Pet owners love their animals, and while they strive to meet their pets' basic needs of food, shelter, and health, they also want to fulfill the pets desire for play. A variety of pet toys exist, but with such a large population that is capable of a sustainable market, technology will continue to be pushed forward in said market.

Cats can be very playful creatures. Even though they are domesticated animals, they still display natural predatory instincts that often manifest through play. They run, they climb, they jump, chase, pounce, and leap, whether it be outside or all throughout their owner's home. Cats love the thrill of the chase. They will chase objects and run from them. KittyBot will be a mobile device. This mobility will engage the animal to play.

Our group consists of two electrical engineers and two computer engineers. Our electrical engineers will give KittyBot life through designing circuits and power systems, while our computer engineers will give KittyBot brains by developing the programs and algorithms that will influence KittyBot's behavior.

2.2 Objectives

The overall goal of this project is to produce a robotic pet toy. The main function of the robot will be to interact with pets, mainly cats. More specific goals for the robot include:

- Durability: This project will need to be durable. The robot is intended to interact with animals which can be, to say the least, unpredictable. Outer casings of aluminum or plastic should have sufficient durability to withstand even the roughest contact with a cat. The main durability concern is the electronic components of the robot. Components such as breadboard circuits, printed circuit boards, and microcontrollers can be severely damaged by cats clawing and biting them. These components need to be protected.

- Maneuverability: The robot will need a concern degree of maneuverability. The robot will primarily operate in an indoor space. The robot will move across common household flooring surfaces such as wood, tile, and low carpet.

- Size: The robot is meant for indoor use with household cats. Because of this, the size of the project has to be kept to dimensions reasonable for this sort of environment. The robot should not exceed 60 cubic centimeters in overall size.

2.3 Requirements Specifications

This section will go over the requirement specifications of this project. These requirements detail what KittyBot needs to be capable of in order to be successful. The appropriate values and constraints will be detailed for each requirement in order to properly access said requirement.

2.3.1 Structural Requirements

This section details the requirements and specifications regarding the structural and physical aspects of the project. The size and weight of every element must be carefully consider in order to keep within the desired small form factor. A key component is the motors. The motor should be small enough to within a central compartment or chassis. They should also not be too heavy.

Next is the chassis itself. The principle design for this project is a spherical robot design. This means it will need an inner chassis to which the motors and all other essential electronics will be mounted, as well as an outer spherical shell. The outer shell will determine the overall size of KittyBot, but it will also dictate what the sizes of all the other components need to be because they all have to fit within the outer shell in a reasonable fashion. The inner chassis should be the second largest single component of KittyBot, and will have to be small enough to fit inside the outer spherical shell, but large enough to hold all the electronic components. In keeping with the interest of maintaining as low a weight as possible this piece should be made from a lightweight material. Two common structural materials are metal and plastic. Both are durable enough for this project and malleable enough to form a shape small enough for this component. A metal like aluminum is a great choice of metal for example. The advantage of plastic however is that will metals like aluminum are more durable and the chances of finding a pre-made piece that suits the projects needs and specifications are certainly much higher than other less adequate materials, it is harder to alter metal in general than it is plastic because it is stronger. In all likelihood we won't find a pre-made piece that perfectly fits our needs and it would need alterations of some kind. That would require sawing, shaving, and machining relatively small parts; a bothersome task. Plastics have the advantage of 3D printing. With 3D printing we won't have to worry about alterations because we can create fully customized pieces to suit our needs. A common 3D printed polymer is acrylonitrile butadiene styrene (ABS). This polymer is sufficient in providing the level of durable needed for the inner chassis, but more important it will allow us to print a piece in the desired shape.

Probably the heaviest single component is the power supply. This project is a mobile platform so it will need to run off battery power. Batteries can come in pre-assembled packs or in single cells. The single cells need to be housed in a battery holder.

Specification	**Value**	**Constraint/Comment**
Maximum Weight	1 kg	This is including all components and accessories assembled into the final product
Maximum Overall Size	60 cm long 30 cm wide	The maximum size the group deemed acceptable for a common household

Figure 2.3.1: Overall Structural Requirements

Specification	Value	Constraint/Comment
Maximum Weight	60 g	The weight of a single motor should not exceed this in order to keep overall weight down
Maximum Overall Size	7 cm long 3 cm wide 3 cm tall	The profile of an individual has to not exceed this in order to fit the center chassis

Figure 2.3.2: Motor Structural Requirements

Specification	Value	Constraint/Comment
Maximum Weight	75 kg	Material could be metal or plastic to give desired function and weight
Maximum Overall Size	10 cm long 10 cm wide 10 cm tall	The maximum size to keep the inside of the sphere reasonable

Figure 2.3.3: Inner Chassis Structural Requirements

Specification	Value	Constraint/Comment
Maximum Weight	200 kg	Battery pack may reach this level of weight if an external battery holder with wire connectors and on/off switch is used
Maximum Overall Size	7 cm long 7 cm wide	Overall size of battery pack

Figure 2.3.4: Power Supply Structural Requirements

2.3.2 Performance Requirements

This section details the requirements and specifications pertaining to the performance aspects of the project. A major determining factor in the systems performance is the motor. Since a small motor is preferable we can use a lower current. The speed of the motors does not need to be that fast since this is an indoor pet toy. The torque just needs to be high enough to carry the overall load of the entire structure. This project will operate at a relatively low power. To this end, the voltage will be low enough to facilitate this, but not be too low as to where the torque would fall to unacceptable levels.

Specification	**Value**	**Constraint/Comment**
Minimum Battery Life	60 minutes	On a full charge, the system should be able to operate for this long
Minimum Speed	60 cm long 30 cm wide	The maximum size the group deemed acceptable for a common household
Maximum Speed	5 mph	In order to maintain a safe operating speed, this speed should not be exceeded
Minimum Speed	1 mph	The device should at the very least reach these speeds

Figure 2.3.5: Overall Performance Requirements

Specification	**Value**	**Constraint/Comment**
Minimum Torque	2 kg-cm	This is what is needed to move the system
Maximum Torque	10 kg-cm	This amount of torque is adequate for the scope of the project, anymore could possibly be dangerous
Minimum Speed	30 rpm	This is the speed (in

		rotations per minute) a single motor needs to be able to reach
Maximum Speed	50 rpm	This is an adequate speed for the project. Spinning any faster is unnecessary and potentially dangerous
Minimum Voltage	4 V	Baseline voltage required for operation
Maximum Voltage	6 V	The most needed to keep the project in a low voltage range

Figure 2.3.6: Motor Performance Requirements

3 Research related to Project Definition

The following section will detail the research performed for the development of Kitty-Bot. Once the initial concept, goals and requirements were all created the research process of Kitty-Bot could begin. The process began with searching for similar projects and products that already exist. Since Kitty-Bot is intended to be a cat toy, robotic pet toys were some of the first products looked at. One of the principle overall designs of Kitty-Bot is to make it spherical in shape. With that in mind, many spherical robot projects and toys were researched. Information gathered from these existing devices helped to focus the design of Kitty-Bot. Next, the technologies of the individual components that will make up Kitty-Bot needed to be researched. These components are the building blocks of Kitty-Bot, so the attributes of different technologies needed to be accessed in order to determine what would best fulfill the requirements of Kitty-Bot, and what would best work together to create a cohesive whole in the final product.

3.1 Existing Similar Projects and Products

3.1.1 Autonomous Ball Collector

This project, the autonomous ball collector, is from an Engineering team of students here at University of Central Florida. The idea was to make an autonomous ground bot that would detect loose balls and automatically scoop them to make it easier for tennis players to deal with picking up balls. The plastic casing holds the tennis balls and protects the circuitry as seen below in **Figure 3.1.1.** The plastic is durable enough to withstand oncoming tennis balls. KittyBot also needs a strong plastic casing to withstand cats playing with it. Another similarity is that KittyBot will be fully autonomous as well which made this project a good reference for general robotics.

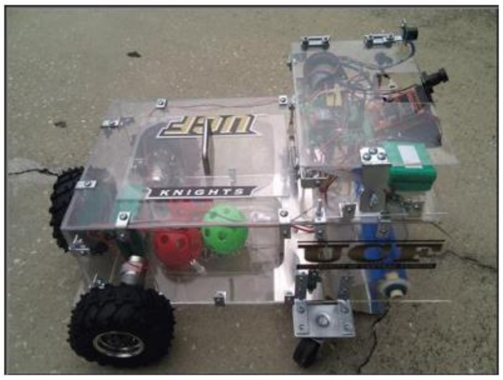


Figure 3.1.1: Autonomous Ball Collector

The Autonomous Ball Collector was made with mindset to convenience the user. The user turns it on and lets the robot do the work while they play tennis. KittyBot will be similar in the sense that a user turns it on and will be something a cat can play with, without the user having to step in and help it get unstuck from corners of rooms.

The project's software interface is an AVR programmer made by Atmel. This Atmel chip is a good reference to look at because it is relatively cheap and easy to use which are good specifications for our KittyBot project. The chip utilizes a flash memory and will execute the program that is written inside. Their chip runs at the speed about 10MHz with built-in 1KB of RAM and 10KB of storage. The idea is to consume the least amount of energy as possible because tennis matches can last over an hour which was one specification this group set out to do.

3.1.2 Puppy Pal

The Puppy Pal is a senior design project done in 2014 by project members Scott Smith, Afzal Schafi, Anson Contrares, and Cameron Riesen. Their project is similar to the KittyBot in the sense that it was made to be an interactive toy with animals. Shown in **Figure 3.1.2**, they have a very similar idea with the round casing to one of the design considerations for KittyBot. The Puppy Pal was created to have a user interface to control the ball with an Android device. The creators wanted to add additional components to the inside like LEDs and an amp. Flashing lights and random sounds were creative ideas in coming up with other ways to attract an animal’s attention to play.

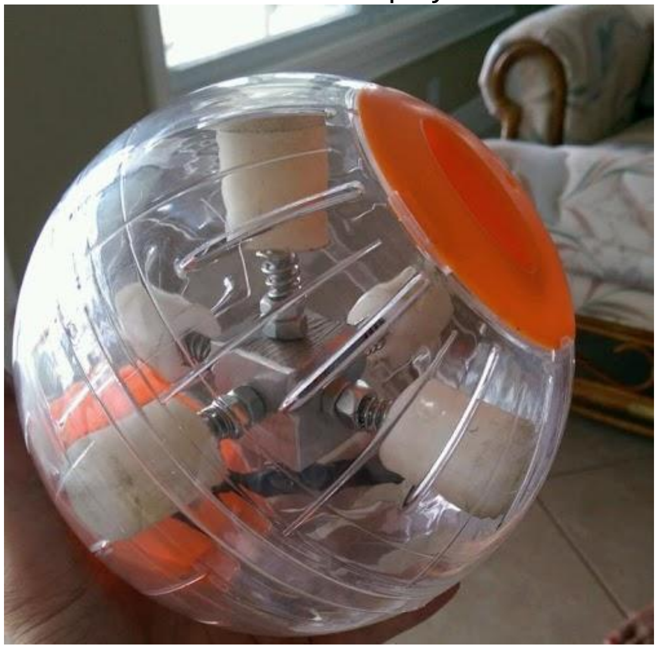


Figure 3.1.2: Puppy Pal

The intended function of KittyBot is to be a robotic cat toy, so we looked at similar products that are current in the market. A common type of product that relates to our project is a motorized chase ball. This type of product is made and sold by several companies. The product mainly consists of a battery-powered motor encased in a plastic ball that can be separated in half down the middle.

Attached to the outside of the ball is usually a tail coated in synthetic fur so that it resembles a small furry animal. Once turned on, the motor inside the ball rotates causing the ball to begin rolling on its own. As the ball rolls the tail flips and flops around along with it. The rolling ball along with the erratic movement of the tail are meant to engage the cat in play. As stated prior these are common products, but we have also researched a few more specific products developed by their own companies. Here is a more in-depth look at them.

3.1.3 Hexbug

Hexbug is a company that sells toys and robots ranging from small R/C toys aimed at children to larger, more complex robots for builders and hobbyists. They also spot a line of electronic cat toys. They currently have a few designs. First is the Hexbug Nano Robotic Cat toy which is a small robot designed to look like an insect. The Nano has five legs on each side and scurries around, mimicking a bug’s movement. There is also a furry tail attached by a string to the back of the Nano to entice cats to chase it. Hexbug’s other design is the Hexbug Mouse Robotic Cat toy. This product is a bit bigger than the Nano cat toy and as its name would suggest it is decorated to look like a mouse. This toy comes in two variants, a remote-controlled version and a fully robotic version. The products are shown in **Figure 3.1.3.**

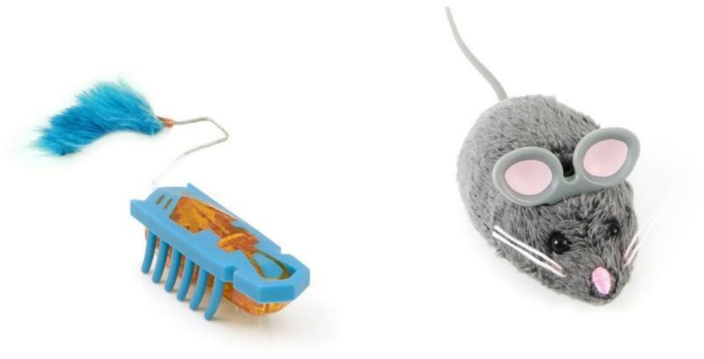


Figure 3.1.3: Hexbug Nano and Hexbug Mouse

(From hexbug.com)

3.1.4 Rotundus GroundBot

Since our robot has a spherical design, we researched other spherical robots. One product we looked at was the Rotundus GroundBot. GroundBot is a robotic mobile platform with a spherical shape and two cameras on the sides of the sphere. It is primarily a mobile surveillance platform intended for use at large secure locations such as airports, warehouses, harbors, and power plants. The spherical and robust design allows for GroundBot to better traverse the rough terrain some of these locations can have. GroundBot can be remote controlled or set to a path using GPS. Internally the GroundBot has a pendulum attached to a motor. The motor moves the pendulum arm. When the pendulum leans in a certain direction the center of gravity of the sphere shifts. This causes the sphere to roll in that direction. The motor keeps the pendulum arm up in a certain direction which allows for continuous movement. **Figure 3.1.4** shows the GroundBot and the internal schematic of how the pendulum arm mechanism works.

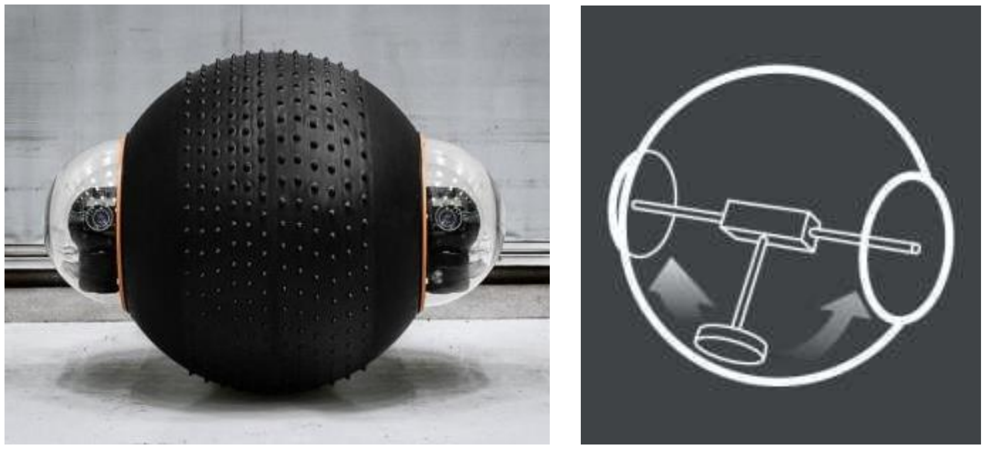


Figure 3-4: Rotundus GroundBot and Internal Schematic

3.1.5 Sphero

Sphero is a robotic ball toy. Owners control the toy's motion and the LED color displayed with a smartphone application. Sphero uses Bluetooth communication to receive its commands. In addition to the basic control app, Sphero's creators developed a series of programming environments to encourage Sphero owners to be creative and make their own apps. Some of the apps let the user create a path for Sphero to follow. To track movement, a three-axis accelerometer and a gyroscope were installed. One impressive feature is Sphero's ability to charge wirelessly.

KittyBot is heavily leaning towards the spherical shape design. The great variety of apps developed for Sphero shows that there are many applications for this type of toy. At first thinking about how the KittyBot will function and move autonomously was difficult to picture. Sphero has given us some guidance towards the first step in making KittyBot autonomous. **Figure 3.1.5** shows the Sphero.

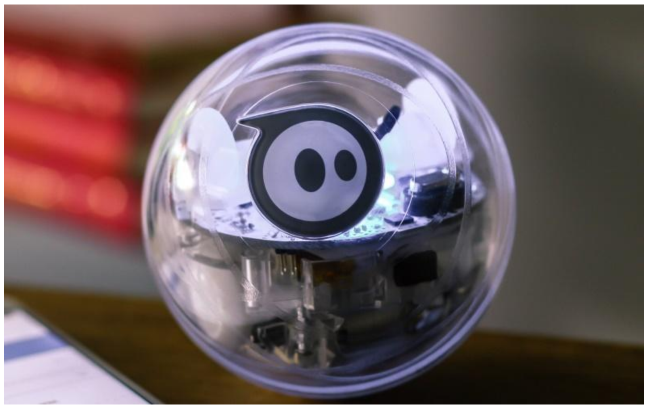


Figure 3.1.5: Sphero

(From www.sphero.com)

Another fun product on the market is the Sphero 2.0, made by the company of the same name. Sphero 2.0 is their latest model and it is a small spherical robot, about 7.5cm in diameter. It is incased in a sealed plastic shell. This makes Sphero waterproof, giving it the ability to traverse bodies of water. The highlight feature of this product however, is its ability to work in tandem with smartphones. It connects with smartphones through Bluetooth allowing user to interact with Sphero through a plethora of smartphone applications available on digital marketplaces. These range from changing the lighting of Sphero's LEDs, directly controlling Sphero's movements, or pre-programming directions for Sphero to follow. A scaled up version of this toy can be seen in BB-8, the android character featured in the 2015 film Star Wars: The Force Awakens. Sphero is responsible for both the robot used in the films and the mass-produced toy versions of BB-8.

BB-8 has a stationery head that sits on top of the rolling ball as it moves. This is because the ball proportion has an internal gyroscope. The gyroscope helps in maintaining stability in the ball as it moves.

3.1.6 Remote Controlled Basketball Robot

In this project, a basketball moves along the ground based on input from a two-channel radio and receiver. Inside of the basketball, a hamster ball holds the chassis. The sides of the chassis are attached to the outside of the ball. This holds the chassis in the middle of the ball. A drive motor, servo, and steering arm are used for drive and steering. A gyroscope was also included.

The bottom of the steering arm holds the batteries and weights. With enough weight, the chassis is kept parallel to the ground despite the ball's motion. As the motor and servo rotate the steering arm in the desired direction, the rest of the ball is pulled forward to keep up with the new center of mass. The gyro senses the changes in rotation, allowing for control. **Figure 3.1.6** shows an internal schematic of the project.

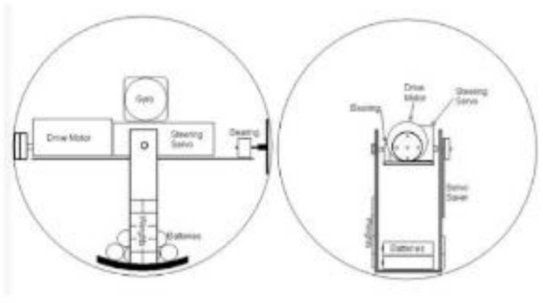


Figure 3.1.6: Remote-Controlled Basketball Schematic

This robot's mechanical system could be a straight-forward solution to the mechanical design of KittyBot. The main obstacles of this design are the weight used, the construction or purchase of the chassis, and control. The use of only a gyroscope in this project left much to be desired in precision. If KittyBot had used this method, it would also include an accelerometer to achieve a more clear-cut sense of maneuverability than the basketball robot.

3.1.7 Product Research Conclusions

The myriad of products researched gave great insights into the design of Kitty-Bot. After researching these robots and toys, the spherical design became much more preferred. Comparing the main two spherical products researched, Sphero and GroundBot, elements of both provide good inspiration. When it comes to their internal movements designs, Groundout's pendulum arm design is an elegant solution to spherical movement. Sphero's innards are more akin to a scooter stuffed in a ball. Rotating wheels and gear shafts of the internal car-like unit move inside the spherical casing causing it to roll. Sphero's design falls more in line with the desired design of Kitty-Bot. It is a toy, which means it is a smaller scale project. GroundBot is a much larger orb-like robot than Sphero and the intended size of Kitty-Bot. It is meant to be an all-terrain robot that can cover large distances. Sphero and Kitty-Bot are meant for use in the small controlled environments of homes. Sphero's internal car design is more applicable in this smaller scale than GroundBot's pendulum.

As a pet toy, Hexbug's line of toy robots greatly inspire Kitty-Bot. These products give insight into making the devices appealing and eye-catching to pets. Things like the bright colors, lights, and sounds all help to entice the animals to interact with the device. The Puppy Pal greatly resembles how KittyBot is intended to look. The internal mechanisms are very different from what will be considered for KittyBot.

3.2 Relevant Technologies

The following section details the research of the relevant technologies to the primary components of Kitty-Bot. First will be a look at motors, which are foundational because they are the primary movers of the device. That will be followed by an in-depth look at microcontroller. The microcontroller's importance is paramount because it will act as the “brain” of Kitty-Bot. It will direct all of Kitty-Bot's motions and actions. Lastly, power supply research will be examined. The power supply's importance is obvious; it will give Kitty-Bot the power to function.

3.2.1 Motor

An essential piece of technology for this project is a motor. A motor will be needed to move the robot. A few different types of motors have been researched in order to determine which would be the best fit for the project. Some motor types under consideration are stepper motors, direct current (DC) motors, and servo motors.

The stepper motor is a very precise motor. It can allow for sharp starting, stopping, and reversing. Stepper motors also tend to run cheaper than the other types of motors under consideration. There are several advantages that the stepper motor may provide KittyBot. One advantage is that the stepper motor is extremely meticulous in calculating its motion. The extreme accuracy of the stepper motor allows for immediate acceleration and deceleration, both forwards and backwards. The stepper motor provides a strong level of control to the user. Another advantage of the stepper motor is that it is inexpensive when compared to the servo motor and the direct current motor. However, there are certain drawbacks to the stepper motor. There are drawbacks to the stepper motor. Firstly, they are slow. They are also noisy. It could potentially scare animals with its excessive noise. The servo motor and the direct current motor both provided a higher rate of acceleration when compared to the stepper motor. Our team felt that this may cause a problem if KittyBot was found being used in an outside environment. However, being that KittyBot is only meant for indoors, our team concluded that this drawback was not crucial to its success. A drawback to the stepper motor that could hinder the usefulness of KittyBot was its noise level. The stepper motor is the loudest motor when compared to the other two, which might frighten the animal that is trying to play with the device.

DC motors allow for higher speed continuous rotations. Such high speeds may be excessive for this project. KittyBot is meant to be an indoor cat toy, so very high speeds are unnecessary. Servo motors generally seem to be a happy medium between DC motors and stepper motors. They offer more precise movement than DC motors. They produce less noise than stepper motor and can reach higher speeds. A drawback of the servo motor is that it has a limited rotational range. This can be tuned however. The first characteristic our team noticed was that the servo motor was not as loud as the stepper motor. The servo motor was also able to function at a faster speed than the stepper motor.

When it came to comparing the movement of the servo motor with the stepper motor, there was very little difference in timing. The stepper motor proved to be more precise than the servo motor. As a team, we concluded the difference to be trivial. There was one disadvantage to the servo motor that was unnoted with the stepper motor: rotational range. In the end, after researching whether or not our team could fix this flaw within KittyBot, we concluded that a simple tune-up would suffice.

Lastly, our team analyzed the findings of the previous motors to those of the direct current motor. The direct current motor was lower in volume when compared to the stepper motor, and was around the same decibel level when compared to the servo motor. The direct current motor was the fastest motor out of all three motors. Our team found that the major difference between the direct current motor and the other two motors was the precision in its movement. The direct current motor was recognizably slower in its timing.

Our team felt that accuracy in timing and detection was crucial to the success of the project. Therefore, the direct current motor was eliminated as a possibility. When deciding between the stepper motor and the servo motor, our team decided that the servo motor provided a nice balance between precision and noise level. Although the stepper motor was slightly more accurate, we determined the miniscule difference to be negligible. In the end, our team chose the Parallax Standard servo motor.

3.2.2 Microcontroller

The microcontroller is responsible for controlling the entire system. When selecting one factors such as processor speed, memory capacity, power consumption, and number of available ports must be considered. Besides the microcontroller’s specifications, its cost must also be taken into account.

A plethora of considerations came into play when deciding what microcontroller to use. This small computer will hold the processing power memory, programmable in and outputs and essentially be the brains of our entire system. Also, the microcontroller will be the base of what will eventually become our printed circuit board. Without this vital piece instructions cannot be compiled into an executable logic that will execute our algorithm. We were mindful of things but certain things were less important than others. For example, processing speed might be a vital nerve for a gigantic system. However, keeping in mind that our system will be relatively simple processing power will not be something that greatly consider. For the same reason, memory wasn't something that was of grave importance. Power consumption on the other hand was a considered a pillar to our success. Our device does not have allot of breathing space to be abundant with space or energy. Being that everything has to fit into a spherical container we need the entire system to consume as little power as possible and also take up as little space as possible. The number of digital pins however was important for us and was a factor of importance for our consideration. Analog pins were not an important factor being that we don't at this stage believe that we use the analog side. That being said having good analog to digital converters would be a useful tool to have in our back pocket should we need it. Lastly, how the output ports were organized for motors and what software we would consider to reconcile the two was of grave importance for us.

Many different microcontrollers were researched. The following figures will list the microcontrollers as well as their pros and cons.

Arduino UNO Rev3: $25	
Pros	**Cons**
● Cheaply Priced ● Native IDE ● Programmable in C, C++	● Slow Clock - 16 MHz

Figure 3.2.1: Arduino UNO Rev3 Pros and Cons Table

BASIC Stamp 2: $49

Pros	Cons
● Small Physical Size ● Low Power Consumption ● Native IDE ● Programmable in C, C++ ● PBASIC Language	● Slow Clock - 20 MHz ● Small Memory Capacity - 32-byte RAM

Figure 3.2.2: BASIC Stamp 2 Pros and Cons Table

BeagleBone Black: $55	
Pros	**Cons**
● Fast Processor - 1 GHz ARM Cortex Processor ● Large Storage Capacity ● Native IDE ● Compatible with Android, Debian, etc.	● Requires knowledge of Linux ● High Power Consumption - 5 V at 200 - 450 mA

Figure 3.2.3: BeagleBone Black Pros and Cons Table

Raspberry Pi: $40	
Pros	**Cons**
● Fast Clock - 700 MHz ● Programmable in multiple languages ● Very impressive quad core processor ARMv7 ● Possible to integrate Ubuntu and other Operating Systems	● High Power Consumption - 5V at 2 A ● Requires knowledge of Linux ● Designed to process video and audio more efficiently than other microcontrollers. Could be a pro depending on what we want.

Figure 3.2.4: Raspberry Pi Pros and Cons Table

TI Tivia C: $13	
Pros	**Cons**
● Very inexpensive ● Energia - An Arduino based IDE ● Compatible with Texas Instruments software libraries	● Less native libraries than more commonly used microcontrollers

Figure 3.2.5: TI Tivia C Pros and Cons Table

TI MSP430: $12 - $24	
Pros	**Cons**
● Very inexpensive ● Energia - An Arduino based IDE ● Low Power Consumption - 1.8 - 3.6 V at 200 uA	● Needs peripheral for prototyping ● Relatively low memory

Figure 3.2.6: TI MSP430 Pros and Cons Table

Wiring S: $35	
Pros	**Cons**
● An abundance of AD pins and PWM outputs for sensors and motors. ● Allot of memory SRAM and Flash to store code and increase process speed	● More memory than needed ● Very little documentation and not much of an online community for problem solving support ● Very high operating voltage

Figure 3.2.7: Wiring S Pros and Cons Table

990.005 MuIn - Multi Interface Board with PIC18F2520: **$35**	
Pros	**Cons**
● Many components, ADC Channels, GPIO/SERVO, EE/EXPORT/USB SOCKET, LEDs, I2C BUS ● Up to 12 sensors can be connected ● Can attach more motors than other devices	● Very little documentation and not much of an online community for problem solving support ● Native Instruction Set

Figure 3.2.8: 990.005 MuIn - Multi Interface Board with PIC18F2520 Pros and Cons Table

Given the pros and cons of each microcontroller based off of their technical specifications and the group's prior experience with some of the systems in question, the TI MSP430 was chosen. The MSP430 has low power consumption which will allow for extended operating times and longer periods of interaction with pets. Also with the MSP430 consuming less power, KittyBot will be under less stress. Considering that our motors need to be able to move the weight of the microcontroller, power supply, and spherical chassis; weight must be cut down wherever possible to ensure proper functionality. The MSP430 also supports many peripherals. Interfacing Texas Instruments brand peripherals will be easier because of greater compatibility. The most important reason TI MSP430 was chosen is our familiarity with the hardware. Similar features can be found in some of the other microcontrollers, but our past experience with the MSP430 will make it easier to use for the group.

3.2.3 Power Supply

KittyBot's power supply is will consist of two or four AA batteries.

Component	Input Voltage	Current

Microcontroller	3.3V	330 μA
Motors	5V	90-190 mA
Sensors	5V	22 mA

Figure 3.2.9: Power Overview

Batteries

Four types of batteries were researched: nickel cadmium, nickel metal hydride, and lithium ion batteries and alkaline batteries. The kitty-bot will run on 2-4 AA sized batteries and be able to run for at least an hour nonstop. The battery pack will sit on top of the PCB board within the KittyBot ball casing.

The batteries will need to be able to supply enough power to all the components. The project is small and compact so the goal is to keep it as low power as possible. The servo motors use about 5 volts and 90 mA to 190 mA each. The batteries will need to be able to supply power also to the proximity sensors which use 5 volts and draw around 12 mA of current.

NiCd

Nickel-cadmium (NiCd) used to be the cheapest of the batteries, but a rise in popularity of NiMH and lithium ion batteries caused them to become cheaper and closer in price. The NiCd battery has highest current output among the three types but is outperformed by NiMH and lithium ion batteries. NiCd takes around one to two hours to fully recharge. The battery needs to be fully discharged before recharging to prevent memory effect in which the battery holds less charge over time.

NiMH

Nickel-metal hybrid (NiMH) batteries are a newer technology than NiCd. It's known to have limited memory effect and higher energy density than NiCd. NiMH battery has good current output and is more environmentally friendlier than NiCd. Rechargeable NiMH batteries are available in 1.2 V per cell. An advantage NiMH has is its ability to discharge a constant 1.2 V until the battery is depleted.

Lithium ion

Lithium ion batteries are the most advanced technology among the three. It is environmentally friendly and has the same energy as NiMH battery but weighs less. Li-Ion has higher energy density than the other two rechargeable batteries. A great advantage lithium ion batteries have over NiMH is that they have no memory loss effect. Rechargeable Li-Ion single cell's nominal value is 3.6 V, which means it would require three times as many nickel batteries. If lithium-ion batteries are used, a protection circuit must be created to use them safely.

Alkaline

Alkaline batteries were researched as well to have an alternative possibility other than rechargeable batteries. Even though it's the only disposable battery here, the alkaline battery has 4 times the capacity of NiCd or NiMH. The nominal cell voltage for an alkaline battery is 1.5V. Alkaline disposable batteries are readily available, commonly used, and cheap. Some negatives about alkaline batteries are that they are heavy and have low power capacities.

Battery Selection

The battery selection came down to two choices in the end. The lithium iron phosphate($LiFePO_4$) batteries or the alkaline batteries. We compared the two to figure which one we would choose. Since the alkaline batteries provide 1.5 volts each, a pack of 4 would give around 6 volts, and we measured it to provide 5.3 volts which was well within the range for our components. Two lithium batteries would provide 6.4 volts but we measured it to be 5.5 volts which is the maximum acceptable voltage for the voltage regulator, not the motors. Since the pendulum counterweight is mostly the battery pack it proved to roll better with more batteries to stabilize the chassis that's holding our circuit and motors. The current draw from KittyBot is only 600 milliamps and each battery provides over 1200 mAh which would give KittyBot at least 2 hours of run time. This meets our goal set for run time of at least one hour. In table 3.2.11 the two batteries were compared using specifications important to KittyBot's design.

Specification	$LiFePO_4$	Alkaline
Price		✓

Weight		✓
Size	✓	
How long they last	✓	✓
Efficiency		✓
Overall		✓

Table 3.2.11 Used to show which specifications were more desired in each battery

Overall the 4 alkaline batteries were a better selection than the 2 lithium iron phosphate batteries for KittyBot. The biggest factor was the weight for the pendulum. On more frictional surfaces like carpet, the inside chassis sometimes spins itself once to create momentum to start rolling the hamster ball. If the weight was lighter it would be slower to start gaining momentum.

Specifications	Lead Acid	NiCd	NiMH	Li-ion		
				Cobalt	Manganese	Phosphate
Specific energy density (Wh/kg)	30–50	45–80	60–120	150–190	100–135	90–120
Internal resistance[1] (mΩ)	<100 12V pack	100–200 6V pack	200–300 6V pack	150–300 7.2V	25–75[2] per cell	25–50[2] per cell
Cycle life[4] (80% discharge)	200–300	1000[3]	300–500[3]	500–1,000	500–1,000	1,000–2,000
Fast-charge time	8–16h	1h typical	2–4h	2–4h	1h or less	1h or less
Overcharge tolerance	High	Moderate	Low	Low. Cannot tolerate trickle charge		
Self-discharge/ month (room temp)	5%	20%[5]	30%[5]	<10%[6]		
Cell voltage (nominal)	2V	1.2V[7]	1.2V[7]	3.6V[8]	3.8V[8]	3.3V
Charge cutoff voltage (V/cell)	2.40 Float 2.25	Full charge detection by voltage signature		4.20		3.60
Discharge cutoff voltage (V/cell, 1C)	1.75	1.00		2.50 – 3.00		2.80
Peak load current Best result	5C[9] 0.2C	20C 1C	5C 0.5C	>3C <1C	>30C <10C	>30C <10C
Charge temperature	–20 to 50°C (–4 to 122°F)	0 to 45°C (32 to 113°F)		0 to 45°C[10] (32 to 113°F)		
Discharge temperature	–20 to 50°C (–4 to °F)	–20 to 65°C (–4 to 49°F)		–20 to 60°C (–4 to 140°F)		
Maintenance requirement	3–6 months[11] (topping chg.)	30–60 days (discharge)	60–90 days (discharge)	Not required		
Safety requirements	Thermally stable	Thermally stable, fuse protection common		Protection circuit mandatory[12]		
In use since	Late 1800s	1950	1990	1991	1996	1999
Toxicity	Very high	Very high	Low	Low		

Figure 3.2.12

Lithium ion batteries provide the most energy and are rechargeable and are the best option for the kitty-bot's design. Since the battery pack is small and can only fit two batteries there needs to be most possible energies from the ones researched. There's a clear advantage that the lithium ion battery has over the NiMH battery, the nominal voltage is 3 times as much. The small casing everything is fitting in causes another design constraint due to the size of the housing, although lithium-ion batteries are the superior batteries to NiCd, NiMH or alkaline batteries.

3.2.4 Protection Circuits

A protection circuit is important to protect your power supply and all the components from either a power surge or a component is drawing more current than the power provided. Fuses act as a short when too much current is running through a system. The fuse triggers when there's too much current in a circuit. Once the current burns through its seal, the fuse blows and cuts off the current from going back into the power supply. The fuse prevents current from going through the power supply which is connected to the motor, microcontroller, and PCB, and protects them from becoming damaged.

Another good way to protect components is with a protection diode. The safety diode protects the circuit or device from the harm that a reverse voltage or current can cause. The protection diode only allows current to flow in one direction. When a diode is placed in parallel with a component, as shown in figure 3-6.1 it is typically placed in reversed bias mode because of current potentially flowing backwards through the circuit. Figure 3-6.1 shows the second arrow which is the reverse current flowing through the diode, instead of possibly damaging the motor. The first arrow is correct current flowing into the motor.

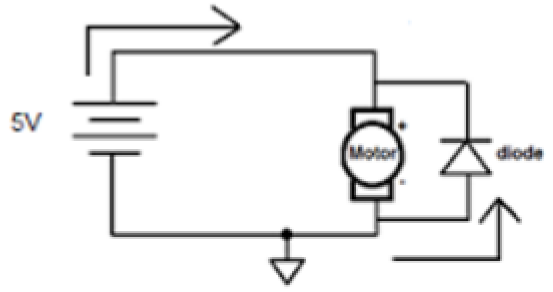


Figure 3.2.13

3.2.5 Over-current Protection

Another important feature for a motor controller is the fact they provide protection to the SERVO motors. While testing KittyBot or an autonomous vehicle in general it is important to protect your components so you can work on testing it rather than waiting to replace fried parts. Components can be protected with the motor controller rather than be exposed to overheating during testing. Heat overloads, shorts and over-current faults can occur for a number of reasons and it's highly valuable to stop the motor controller before damage can be done to itself or the connected circuitry. The device must have current limits set on the internal field effect transistors. If a fault of any kind is detected, a surge protector will be able to prevent damage to the motor controller and power supply. The feedback will cause the circuit to be open disallowing the motor to pass too much current through and protecting the circuit.

3.3 Strategic Components

This section will detail strategic components that can be attached to Kitty-Bot. The content seen below will delve into the components outside of the main components needed for the project. These devices will serve to refine Kitty-Bot's abilities and functionality to allow for a better product.

3.3.1 Communication Hardware Considerations

Remote Control

KittyBot is first and foremost a robot. We want it to be autonomous and make decisions based off of inputs from its sensors. Another option for KittyBot to give it more flexibility as an entertainment item is to add some form of remote control. While operating autonomously KittyBot can fully engage with the pet, but the owner is limited to only the role of a spectator. With a remote control option KittyBot changes from merely being interactive for the pet and observatory for the owner to being interactive for both. Remote control would bridge the gap between pet and owner allowing for an interaction between the two.

There are a few ways to achieve remote control with KittyBot. The first would be through a transmitter and receiver kit. The receiver would connect to the internal central unit of the system while the user would transmit commands with the RC transmitter. Many transmitter/receiver set are available. All that is necessary is a 2-Channel transmitter is all that would be necessary, one for forwards and backwards and the other for left and right rotation. The most common style of transmitter is the two joystick design; the left joystick controls front and back movement while the right stick controls left and right rotation. An example of this design is shown in **Figure 3.3.1.**

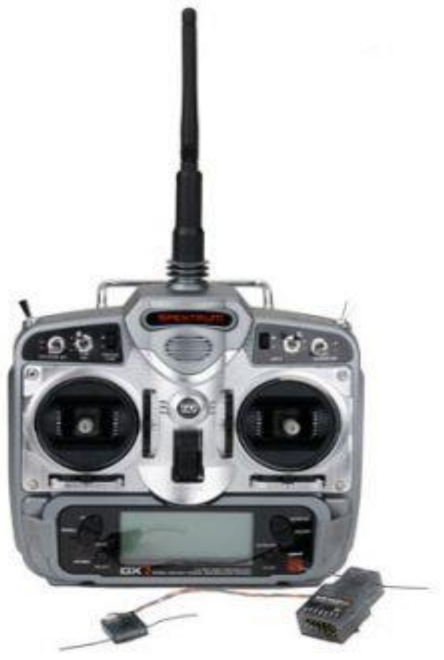


Figure 3.3.1: Spektrum DX-7 Transmitter

The Spektrum DX-7 shown has quite a bit more features than necessary, but it displays the style of controller we would consider, the common dual-joystick style.

Another option for remote control is through Bluetooth, more specifically Bluetooth control through a smartphone application. A Bluetooth module can be attached to the central control unit to allow for communication between the device and a smartphone or tablet. A module like the HC-05 or HC-06 could accomplish this. The HC-05 (**Figure 3.3.2**) is capable of slave or master (receiving or transmitting) settings while the HC-06 is only capable of slave settings. They are both physically the same measuring only 3 cm long. Oddly the HC-06 is normally not cheaper than the HC-05.

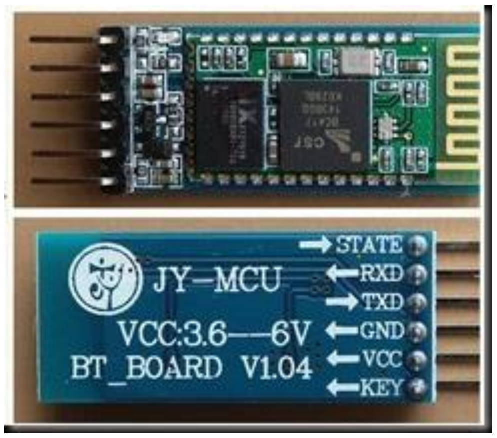


Figure 3.3.2: HC-05 Bluetooth Module

The HC-05 would have to work in tandem with a smartphone application. This comes with many choices. The first would be which mobile operating systems to develop for, Apple IOS, Google Android, or Windows Phone. While slowly growing, the share of the mobile phone market Windows holds is very small; and while their Windows 10 integration on their very successful line Surface tablet computers would allow for an increased plethora of choice in programming languages, IDEs, etc., we feel the best course of action for the Bluetooth control option would be to focus on smartphones as the primary transmit device and let tablets be secondary. Apple IOS is a much more widely used operating system. The fact that it is limited to the Apple iPhone line of smartphones is not an issue because of the brands staggering popular and widespread use. Market and familiarity would not be a problem. When it comes to programming the options become quite a bit less. IOS apps have classically been mostly coded in Objective C, which is a very robust language fully capable of delivering the desired results. Apple has also introduced the Swift programming language to allow for a more user-friendly coding experience. iOS would present a challenge with the use of the app. Will the app would be able to be tested on an iPhone in developer mode, the process of trying to get an app on their app store requires a $100 developer fee. The Android operating system is incredibly widespread. It is very popular and used on a multitude of different devices. Android development is mostly done in Java. The other good part of Android app development is getting your app up and running for testing and on the Google Play store is completely free.

The best fit for developing an app would be the Android operating system. The main reason for this is simply familiarity. Our group has much more prior experience developing for Android. We have little to none developing for iOS. While Windows can utilize a wide variety of programming languages, we don't readily have access to a Window Surface or similar product. That also takes away the mobile element we'd wish to achieve. An Android application would allow us to interface with the Bluetooth module inside KittyBot. Nearly all modern smartphones include Bluetooth technology. This would mean nearly all cell phone user have a remote to KittyBot in the palm of their hands. This is a design element inspired by Sphero which uses Bluetooth and a variety of Sphero developed apps to control the robot.

Infrared Communication

Infrared communication is a very popular communication technology for many applications such as remote control, robotics, etc. Infrared communication has been used in applications such as remotes effectively since around the 1950's; it is an effective way to wirelessly transmit data. There are many aspects of infrared communication that can make it a suitable means of transferring data given the right conditions.

It is a very low power option that generally requires no more than a couple of AA batteries. Since infrared is a very popular technology, it is also well established and has existed in the market place for many years now. Due to this widespread market utilization it is also a very inexpensive technology and has many resources on the fundamentals of its operation. The data transfer rate of infrared is not as fast as some other technologies (around 4 MB per second), but that is fast enough for this project and therefore is not an appreciable problem.

Unfortunately, infrared communication also has some serious drawbacks. Due to the high frequency with which infrared operates, about 100-214 THz for low range telecommunications, infrared cannot pass through solid objects such as walls or any other solid material. This means that to effectively utilize infrared technology the receiver must be visible to the transmitter or the signal will be totally reflected and not be transmitted/received at all.

RF Communication

Radio frequency (RF) communication is the most popular form of communication in modern technology today; Bluetooth is actually a specialized form of RF communication designed to operate over short distances. Most RF communication technologies are designed to operate over greater distances than technologies like Bluetooth or Infrared with ranges of up to a few kilometers. RF communication technologies are generally classified by the frequency with which they transmit information; the two that will be addressed here are two frequencies in the ISM (industrial, scientific, medical) frequency band, 2.4 GHz and sub-GHz technology. Instead of separating them into two different categories they will be compared side by side due to many commonalities in their operation as well as to highlight some of their differences.

2.4 GHz

2.4 GHZ technology is the most popular frequency in most wireless internet routers today; it has been the chosen frequency in the ISM band for some time due to the IEEE standards. This frequency has a balance between range and penetrability. If allowed to transmit through free space, it has a range that is higher than a 900 MHZ system and also allows for smaller antennas due to the shorter wavelength of a 2.4 GHz RF signal. Unfortunately, signals are not transmitted in free space so there will be greater transmission loss for a 2.4 GHz signal if obstructions are encountered such as walls and other solid objects. Another important feature of 2.4 GHz technology compared to 900 MHz technology is the data rate associated with each; 2.4 GHz technology allow for higher data transfer rates.

Sub GHz (900 MHz)

Sub GHz technology has become increasingly popular over the last few years for a number of reasons. It is still a part of the unlicensed ISM band making it suitable for industrial applications. It has some of the same advantages of 2.4 GHz technology with some vast improvements. While 2.4 GHz experiences better range in free space transmission, sub GHz technology is vastly superior in transmission through environments where obstructions are encountered resulting in an overall better range for sub GHz transmission. This is due to a number of factors, one of which is path loss. Path loss is a mathematical model describing how much of a signal is lost over a certain distance for a certain wavelength. It is given as:

$$Path\ Loss(dB)=20\cdot\log_{10}[(4\cdot\pi\cdot d)/\lambda]$$

Where d is the distance and λ is the wavelength of the transmitted signal. Since wavelength is inversely proportional to frequency it can be seen that a 900 MHz signal would have far superior range compared to a 2.4 GHz signal. As a matter of fact, a 900 MHz signal would have approximately 2.67x increase in range compared to a 2.4 GHz signal. A 2.4 GHz device would have to increase its power by nearly 8.5 dB to match the range of a 900 MHz signal operating at lower power. A graph comparing the path loss of a 2.4 GHz signal to a 900 MHz signal is shown below in Figure 3.3.3.

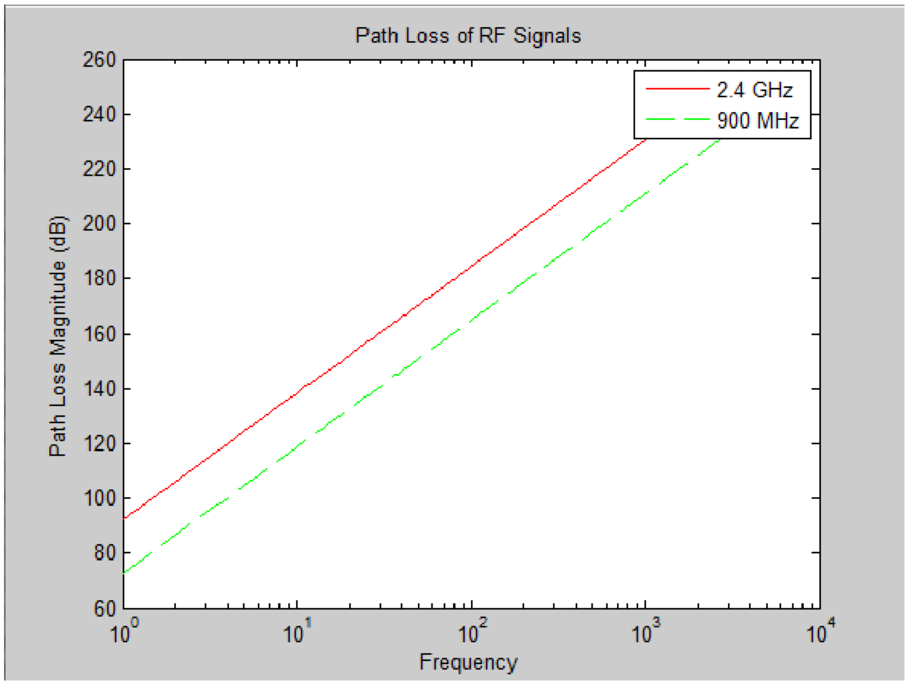


Figure 3.3.3: 2.4 GHz vs 900 MHz

Bluetooth Communication

Bluetooth is another potential design to utilize wireless communication. It is a relatively new technology that has many advantages over infrared and other technologies. Bluetooth was invented in 1994 by the telecommunication company, Ericsson. Bluetooth utilizes radio frequencies in the 2.4 GHz range to transmit data.

As with any technology, Bluetooth has some positive attributes as well as some shortcomings. Bluetooth technology is actually a radio frequency standard that employs a protocol which means that any device operating on Bluetooth technology will operate using that specific frequency range and will also send information in a uniform format. This is one of the biggest benefits to using Bluetooth technology, it can automatically connect. If any two Bluetooth devices are within the operating range and they are both enabled, they will connect and transmit data automatically. This adds a level of convenience for the user since they do not have to worry about formatting how they transmit data or whether or not they are connected.

Bluetooth technology also has the advantage of being a low power transmission option for wireless communication; it transmits about 1 mW per transmission. This would work well with the lower power goal set for KittyBot. Also, Bluetooth can incorporate multiple devices at once due to its frequency hopping which allows up to 79 devices on as many different frequencies communicate with one another. This would be advantageous for this project if multiple robots were to be created and added in the future. Bluetooth is both low power and an easy to use technology. Bluetooth has a limited range of use, but it is roughly 30 meters in all the Bluetooth models that were researched, and the range is long enough for KittyBot.

Although there are many advantages to using Bluetooth technology, there are also some disadvantages that must be taken into account. This would not be a huge problem except that, as mentioned before, this project is supposed to be designed to operate over a substantial distance, ideally much farther than ten meters. Bluetooth is also capable of fast data transfer, up to 2-3 megabits per second which is slower than IR but still fast enough for the for this project.

Decision on Which Type of Communication to Use

A couple Bluetooth chips were researched and many of the specifications were compared such as: cost, operating voltage, size, and the complexity of interfacing the module to the system. Figure 3-1 provides a table comparing the different specifications of potential Bluetooth modules. Observing the table, it appears that all the modules require around the same operating voltage to run. The module RN-42 has a potential of 6 volt operating voltage due to the ability to change its max data rate. The largest module appears to be the HC-06 Bluetooth module, while the RN-42 and the WT11i are approximately the same size. Size is important due to the fact that the components all need to be enclosed in a small round casing. The table shows that the signal distance specification. The HC module has the lowest standard in comparison of the other two modules with 30 feet, the RN-42 ranging from 50 to 60 feet, and the Bluegiga WT11i with a significant line of sight range of 328 to 984 feet. This feature is not as important for KittyBot as it is an inside cat toy and if a controller was made for KittyBot, the user would most likely be in a 30 feet range. The max data rate value for all three potential Bluetooth modules are around the same of around 2 to 3 Mbps. The HC-06 module will be the cheapest of the 3 modules, with the RN-42 and WT11i costing about 30-40 dollars. The KittyBot needs to be cheap in order to ever be profitable so HC-06 is good for this.

With all the specifications taken into consideration, the HC-06 module would be the best potential choice for KittyBot's hardware in integrating a Bluetooth communication device. This would only be considered for a user control design only. It is the best possible choice with this design consideration because the 30 feet signal coverage is an acceptable range for KittyBot; the project was made to be used in a house or room. The HC-06 module can connect to an Android or iPhone device in a simple way by locating the module from the user's phone and entering a given password to connect. The HC module is larger in size in comparison to the other two, but can be negated due to the price of the HC module being much lower.

Bluetooth Module	HC 06	Bluegiga WT11i	RN-42
Operating Voltage	3.3V	2.7 - 3.6V	3.3 - 6V
Size	4.3 x 1.6 x 0.7 cm	35.75 x 14.5 x 2.6 mm	38 x 17 mm
Signal Distance	30 ft.	328 -984 ft. (L.O.S.)	50 to 60 ft
Max Data Rate	2.1 Mbps	2-3 Mbps	3 Mbps
Cost	$10.00	$30.00	$40.00

Figure 3.3.4: Module Comparisons

3.3.2 Sensors

The kitty-bot’s sensors should be able to read and detect objects in its path in milliseconds. The sensors need to be able to detect objects quickly enough to process and send the signal back to the microcontroller and motors to be able to adjust, and roll away to avoid an object. The sensor needs to be reading data continuously. The settling time for the sensors should be in the millisecond range as well when they are powered up.

The sensor should be able to accurately detect obstacles through the balls clear shell, and sense them quickly enough to avoid what’s in the way. The sensor should be able to detect any objects the KittyBot hits and the KittyBot will either be able to avoid it or eventually move away from the object after hitting it. The four sensors researched for all the possible designs for KittyBot were photoelectric(infrared), image(webcam), ultrasonic, and piezoelectric(impact) sensors. Piezoelectric sensors were the most popularly liked among the group. Since KittyBot is continuously rolling the group realized how difficult ultrasonic sensors or photoelectric sensors might be impractical to implement.

Photoelectric Sensors

One type of sensor that was considered is a photoelectric sensor. Since the sensor is in a clear plastic housing, photoelectric sensors will not work with the design because photoelectric sensors can detect transparent surfaces. Photoelectric sensors are useful for other potential kitty-bot designs, they’re able to sense objects in the kitty-bot’s path to a given distance.

The photoelectric sensor is good for detecting a fixed range which is useful for this design. There are 3 main types of photoelectric sensors, through-beam, reflective, and diffuse. Through-beam sensors are the most accurate of the 3 but require a receiver and a transmitter and wait for the light beam between them to break. This won't work in the design because through-beam sensors wait for a break in light rather than project and read what's in front of the sensor.

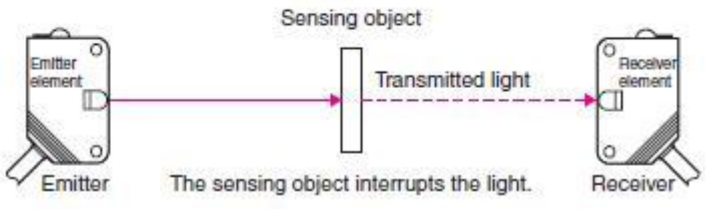


Figure 3.3.5

The retro reflective mode of photoelectric sensors detects objects when the signal is blocked. Retro reflective optic sensors also have a transmitter and a receiver. The transmitter transmits a light beam off a reflective surface across from it. When the light is blocked by a non-shiny surface and the light particles can't make it back through the polarization filter the retro reflective sensor detects an object. This is also not the type of sensor that would good for this type of project because the retro reflective sensor is stationary across from a shiny surface, and the kitty-bot is mobile.

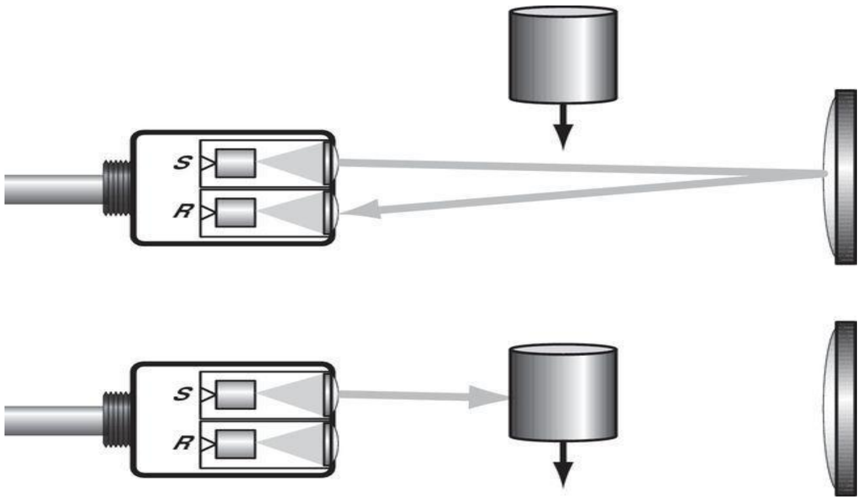


Figure 3.3.6

The 3rd mode of photoelectric sensors is diffused mode. In diffused mode, the transmitter sends out light to an adjustable distance. A receiver reads the light that scatters off the object in front of the sensor and triggers a command. Diffuse mode photoelectric sensors can be used for other designs because it can detect transparent walls or objects. This design requires a different sensor because the sensors need to be protected in a clear plastic casing.

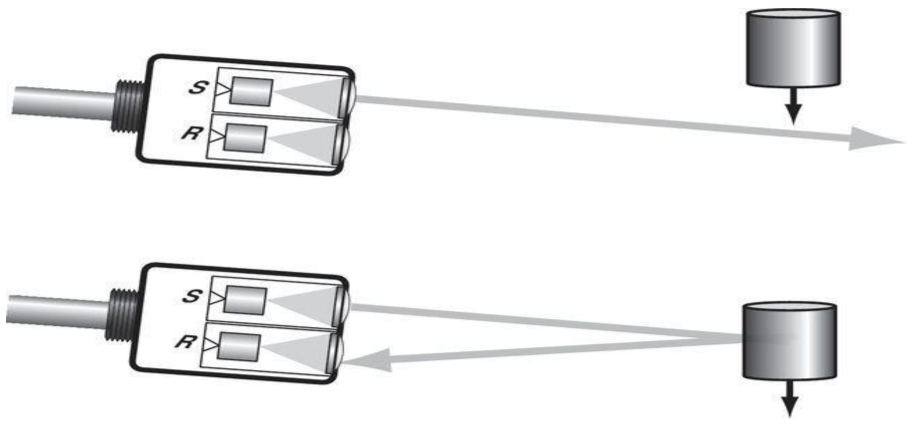


Figure 3.3.7

Image Sensors

Image sensors or webcams take in the light waves from particles bouncing off objects and turn those photons into electrical signals which can be displayed as an image on a screen. Complementary metal-oxide semiconductors (CMOS) are a type of image sensor as well as charged-coupled devices (CCD). The resolution on the image sensors needs to be good enough to have enough pixels to present a clear image for navigation. The kitty-bot will usually be in tight spaces so the image sensor will need to be able to detect and read the objects it’s approaching, quickly and clearly. Since webcams rely on light, darkness could be a design restraint to consider.

CMOS and CCD sensors both take in light through pixels shown in figure 3-4. The sensors take in photons which build up in the highly light sensitive areas to build an image. Using a positive charge, the electrons are separated from the photons. Then the electrons are turned into a tiny voltage which is amplified and can be connected and shown through a screen. For CCD sensors most of the functions are done on the printed circuit board through output nodes. CMOS sensors convert photons to electrons and then to voltage through a diode, all right at the pixels. The diodes send the voltage into a MUX and after the voltage is amplified, an analog to digital conversion is done for the CMOS sensor. This process allows the CMOS sensor to be faster due to the pixels can be processed at once. CCD sensors can only process one pixel at a time through their output nodes so it takes much longer than CMOS sensors. CMOS chips are more low powered than CCD chips, but

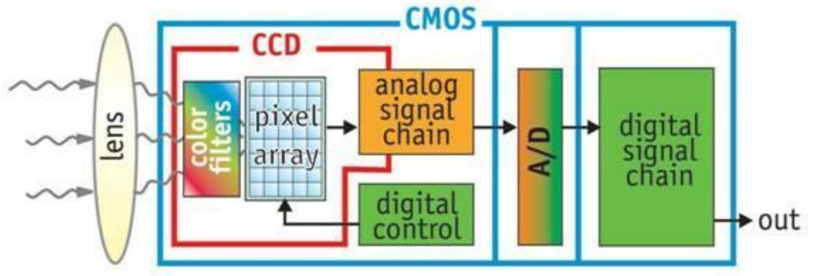


Figure 3.3.8

Ultrasonic Sensors

Ultrasonic sensors use high frequency sound waves to detect items that the sound reflects off from. The sensor keeps track of time for the echo to return, as well as the echo pulse width to determine the distance of the object in the way which is important for KittyBot. Inexpensive ultrasonic sensors usually give off a 40±1000 Hz range of frequency as its sound wave to echo for detection. One popular detection method is SONAR. It is used primarily for underwater because the high frequency waves emitted, it's less than 1 MHz though. SONAR emits echo sound that travels through a medium (air) to detect an object. Upon contact with an object emitted signal is reflected back towards the sensor that listens to reflected sound waves. Reflected signal carries information about direct distance to the object. This presents an ability to obtain a fairly quick response from the object detected.

The ultrasonic sensors use of sound propagation is an advantage for accuracy of distance over infrared sensors. Sound waves are capable of detecting an object regardless of the color so this is good to detect glass surfaces. They are also immune to external disturbance such as vibration, infrared radiation and interference. There will be different scenarios that need to be addressed and resolved by the KittyBot like corners or animals. Another example is when KittyBot detects an object ahead, the sensors needs to detect it and turn away instead of getting stuck in a corner of a house or anything.

The ultrasonic sensors have some drawbacks however that led to the group not wanting to use them. In figure 3-5 it shows the sensor not working due to the angle which is deflecting away the signal and not detecting the wall quickly enough. The theta value for figure 3-5 is less than 45° which is a high value on top of the limited space the sensors would be operating out of. Another reason ultrasonic sensors can be disadvantageous is that they are unable to read small objects in its path as shown in figure 3-6. The ear design for KittyBot would have the sensors elevated and would have trouble rolling past an object long and low to the ground like a branch for example. Another disadvantage of ultrasonic sensors is ghost echo. This is where the sound can bounce off of several objects and result in duplicate and reflected waves with a time interval delay. Sound absorbing materials can also throw off the accuracy and lead to the KittyBot not reading objects before it and getting stuck running into a wall potentially. Ultrasonic sensors are expensive compared to the photoelectric sensors, and our goal is to make it as cheap as possible since KittyBot is meant to be a toy.

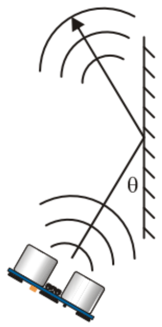


Figure 3.3.9
(From www.parallax.com)

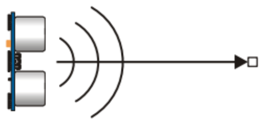


Figure 3.3.10
(From www.parallax.com)

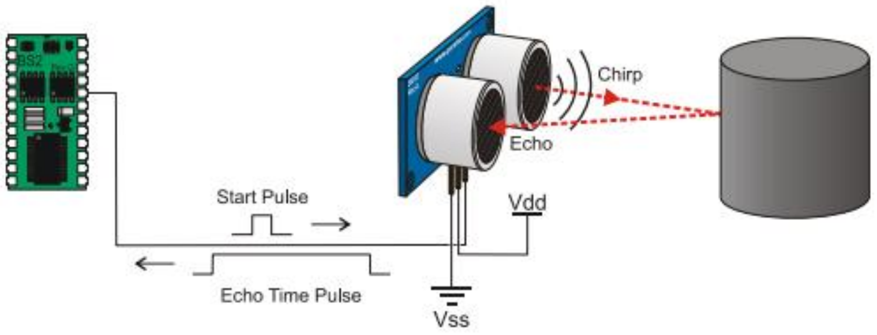


Figure 3-27

Piezoelectric Sensors

Piezoelectric sensors are usually small and very versatile sensor. This sensor is also known as a transducer. The piezoelectric sensor can detect changes in pressure, acceleration, temperature, or force which is important for KittyBot when a cat swipes at the ball it will change direction and roll away. Also the piezoelectric sensor will detect when the KittyBot hits a wall and be able to maneuver away so it doesn't get stuck.

Piezo sensors have their disadvantages however. Since they rely on contact, they may be slow to signal the motors and KittyBot will turn away from objects or animals slower. Another possible design constraint could be that the ball may roll on top of the sensors causing them to trigger. Since the sensors will be placed on the shell casing, which will be moving, wiring these sensors may prove difficult. Based on how they were set up in a video, it may be hard to set the piezo sensors up.

The piezoelectric sensors have their advantages as well. One example is that they're very lightweight and tough so they can withstand the impact of a wall over and over or if an animal that is playing with it. Some useful characteristics of the properties of the sensor are listed below:

- Wide frequency range—0.001 Hz to 10^9 Hz.
- Vast dynamic impact range (10^{-8} to 10^6 psi)

- Low acoustic impedance, close match to water, human tissue and adhesive systems. These sensors can be used to detect signal in muscles and tissues.
- High elastic compliance which means it's useful for KittyBot being able to bend around the round surface of the round casing.
- A high output voltage is generated from a fair impact.
- High mechanical strength and impact resistance (10^9 —10^{10} Pascal modulus).
- High stability—resisting moisture (moisture absorption), most chemicals, oxidants, and intense ultraviolet and nuclear radiation.
- The piezoelectric film sensors can be fabricated into an unusual design like the round shell casing of the KittyBot.
- Can be glued with commercial adhesives so the sensors can be applied easily to the KittyBot.

The idea for using piezoelectric sensors is that the cat would swipe at KittyBot and it would sense the force and react and roll away. The piezo element has a 12mm diameter which will allow them to be placed throughout KittyBot's casing. Some things to consider in setting these up inside KittyBot will be its wiring and making sure they won't trigger while rolling. This piezo sensor provides too miniscule of a voltage when pressed and can't be connected directly to the microcontroller.

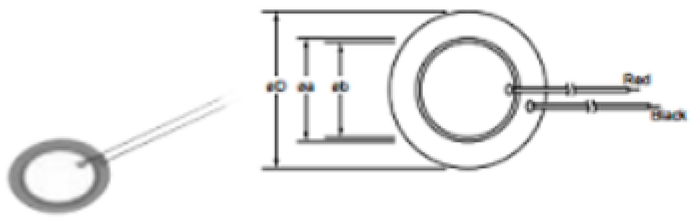


Figure 3.3.11

(From www.sparkfun.com)

One solution to this problem is to connect the piezo sensor to a pnp transistor. The transistor 2N3906 was a good example seen used. The red lead connects to the base, the black connects to the emitter and ground. The collector is connected to the MSP430 to pin 3 like shown below in figure 6.1.4. The sensor provides enough voltage to turn on the transistor which is still not enough voltage to power the microcontroller. In order to get the microcontroller to read the piezo sensor, the MSP430 is set to trigger when it reads low instead of high.

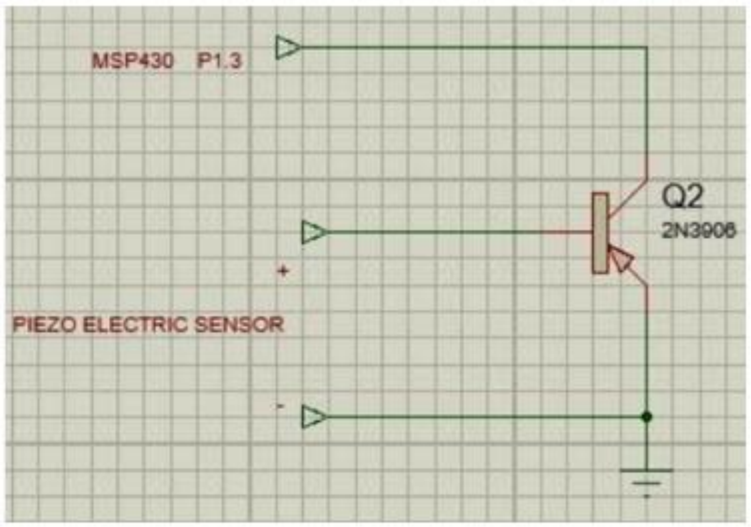


Figure 3.3.12

The previously researched sensors will be used based on the design approach we take as a group. The best approach will be tested and multiple sensors could end up in the final design or it could be just one of the choices. Even though research was done for webcams, it was soon realized that they could not be practical with any of the designs for KittyBot and no part was included.

Sensor Tradeoffs

The sensors that were considered in coming up with this kitty-bot design include: photoelectric, image, infrared and ultrasonic sensors. Each sensor has its positives about it, but the design constraints narrowed down which sensor the kitty-bot would use to detect objects. The shell casing, which allows the ball to roll and protects the circuitry, sensors, and motors, also limits design possibilities, but creativity was needed to make some of the sensors usable. Infrared sensors sense the light given off from objects but can detect transparent surfaces. Infrared wasn't a good option because of the sensors' sensitivity to infrared lights and sunlight. Since the casing is plastic, a small hole would have to be cut in order for the sensor to be usable for this kitty-bot design. I compared the other sensors and looked at the pros and cons (as seen in figure 6-5) to help decide which detection sensor will be used for kitty-bot.

Photoelectric sensors are very good for accuracy of the desired distance of detection. However, these sensors may prove to be difficult to set up with accuracy because of the movement. Since kitty-bot is a toy, it needs to be relatively cheap if it were to ever sell and make a profit, and photoelectric sensors can be pricy. The kitty-bot project is good to use some cheap sensors because it only needs a sensing distance of less than 12 inches. Photoelectric sensors are good because they use a laser or light so they're very fast with detection compared to sound also compared in figure 6-5. Setting a distance to detect 5 inches in front of the plastic ball would be very easy. Laser sensors also read objects in front of it very accurately without being affected a lot by outside factors like brightness, or color of the object. Photoelectric sensors are the worst for power out of the 3 types of sensors researched. The diffuse sensor could be the only photoelectric sensor possible for this project, and for this design, they may be the only sensors available.

Ultrasonic sensors use sound waves to detect items that the sound reflects off from. The housing around the sensors for this design wouldn't be able to work well with ultrasonic sensors because of its inability to send a signal past the plastic casing. Ultrasonic sensors are still a consideration for multiple casing design ideas. These sensors are the lowest power of the 3 sensors as can eb seen in the trade-off chart in figure 6-. These sensors are useful for detecting range accurately, and can detect small objects better than a photoelectric sensor. A big upside to ultrasonic sensors is its ability to detect surface while disregarding brightness, color, or transparency unlike most proximity sensors. The downsides of ultrasonic sensors are that they don't work well with the design and their response time is slower than photoelectric sensors, because light is faster than sound. The ultrasonic sensor also can't detect soft objects that have trouble reflecting noise back well. It's important the sensors stay protected and a casing around them doesn't allow for the ultrasonic sensors to function how they're needed.

The sensor design first considered was an image sensor, or a webcam. This design is practical for the transparent casing around the webcams. The webcam can't be adjusted to sense or see a shorter distance like the diffused photoelectric sensor can. Image sensors are low powered; the ones found online were all less than 500 mW, slightly lower than the photoelectric sensors found. A high resolution camera would be better at detecting objects and colors more clearly but having 2 cameras limits the price and quality of the sensors. The main reason image sensors are ideal for the kitty-bot is because of its flexibility to detect objects through a transparent surface. The sensor needs to be protected and durable to keep the kitty-bot running if an animal were to play with it. Image sensors rely on sensing the colors around it which limits the cameras accuracy to avoid any object in its path.

<u>Sensor Tradeoff Table</u>

Sensor Type	Photoelectric	Image	Ultrasonic
Speed	↑ ↑	↑	↓
Accuracy	↑		↑
Power	↓		↑
Design Flexibility	↓	↑	↓
Weight	↑	↓ ↓	↑ ↑
Size		↓	↑ ↑

Distance	↑	↑	

↑ ↑ = Very positive ↑ = Positive ↓ = Negative
↓ ↓ = Very negative

Figure 3.3.13

3.3.3 Voltage Regulation

Voltage regulators are important in electrical systems. They allow systems to run higher voltages for more powerful components without out sending too much voltage to more sensitive components. The two types of regulators that were researched were the switching and voltage regulators. Voltage regulators don't use much power due to the tiny current that runs through it. The servo motors could require a larger voltage to power them and therefore a voltage regulator will be needed to regulate voltage for the microcontroller and sensors. Having a large gap between V_{in} and V_{out} causes inefficiency in the regulator.

Sometimes motors can draw a huge amount of current that a battery source could not handle. As a result, the whole system could experience a significant voltage drop, causing the microcontroller to reset and not work properly or have the sensor give bad readings. The solution to the problem is placing an electrolytic capacitor parallel to the battery pack. One of the main functions of capacitor is to store large energy quantity during idle periods and give up that energy when other components need it. The higher the capacitance, the more charge it can hold. Many capacitors are labeled with the maximum voltage that the capacitors can handle without damaging them. It is recommended to get capacitors that are rated at least twice the expected voltage drop across them to ensure that they don't explode when fully charged. Since a ceramic capacitor can lose about 50 percent of its capacitance at a rated voltage, it's best to leave a large margin on the voltage rating.

Capacitors are typically connected to the input and output of a voltage regulator. The input capacitors filter out system noise prior to regulation. The output capacitors help the regulator deal with spikes created by the load. The regulator may oscillate at certain temperatures if the capacitors are not present. The large capacitors prevent low frequency interferences and keep the system powered when sudden current surges occur. Small capacitors prevent high frequency disturbances from motors. They have low equivalent series resistance (ESR) that allow them to charge and discharge quickly.

Under faulty conditions such as short circuits and overload, a fuse should be used to protect the motors from excessive current flowing from the battery. The fuse would heat up and blow, therefore, interrupting the current flow and preventing damage to the motors. Time-delay or slow-blow fuses are recommended for inductive loads such as motors. Fast-acting fuses are used for non-inductive loads. Fuse’s voltage rating indicates that the fuse can be used at all voltages not exceeding the rating. An AC fuse can be used on a DC circuit but its voltage should be rated at least twice that of the circuit. Fuses can be connected in series or parallel. If there’re multiple power sources connected in series, then only one fuse is needed to connect in series to the sources and load. If there’re more than one battery connected in parallel, then there must be one fuse for each battery in addition to one main fuse connected to a load. The parallel configuration is obviously less advantageous than the series configuration because it requires a higher number of fuses.

Large capacitors that are fully charged after the robot is turned off can cause components to be accidently shorted and fried. A LED can be used to drain the capacitors and also serves as a status indicator. A dim LED might indicate that the circuit is low in power. The LED should be connected in series with a resistor to prevent the LED from frying. There’re tradeoffs in selecting the resistance. The higher the resistance, the more power it can drain but the LED’s brightness would decrease.

A voltage regulator is needed to regulate the voltage to the microcontroller and sensors. Increasing or decreasing the input voltage even for a fraction of a second would result in the microcontroller resetting or sensor giving bad signals. Even though batteries are specified to operate at a nominal voltage, they are not always at the nominal value. A fully-charged battery can go higher than the nominal voltage. A drained one would drop significantly from the nominal value. Because the microcontroller and sensors consume low current, the wasted power is not significant. As a result, either a linear regulator or switching regulator can be used. On the other hand, motors require a lot of current. In this case, switching regulator is ideal for the motors.

Multiple circuit protection and voltage regulation designs are considered. One design uses two power sources. A battery pack is used exclusively to power the motors. Another pack is used to power other electronics. The design divides the system into two main subsystems. One subsystem consists of one battery pack, motors, and fuses. The other subsystem includes the other battery pack, microcontroller, sensor, and voltage regulator. If one subsystem fails, then it would not affect the other subsystem. However, adding more battery packs means more space is occupied and adding load burden on the whole system, causing the motors to draw more current. Therefore, the design is unfit for our robot. Another design would have only one power source. There is one major drawback of this design, however. Since the whole system is interconnected, a component's failure may affect the other components. For this design, several voltage regulators are taken into account including boost/buck, boost, and LDO (low drop-out) regulators.

.

Boost Regulator Circuit Design

Figure 3.6 shows a circuit design with a TI TPS61232 boost converter. TPS61232 is preferred over the TPS61230 and TPS61231 because it allows a 5 V fixed output voltage whereas the other two have adjustable outputs which require additional resistors to adjust the voltage. The converter is ideal for our project because its maximum efficiency is 96 percent. With an input ranging from 2.3 V to 5.5 V it's able to regulate a fixed output voltage of 5 V that is recommended by the sensors. The converter also delivers up to 2.1 A of current with a 5 V output and 3.3 V input. This current is more than enough to power the electronics in the system. The converter has additional built-in features including output over voltage and thermal shutdown protections, power good output, and power save mode for light load which typically consumes 1.5 uA. The converter is optimized for a one-cell Li-Ion battery, which is usually rated at 3.7 V. Either a single Li-Ion cell or three NiMH cells with a voltage of 3.6 V can be used. C1, C2, C3, L, and R2 are required external components whose values are specified in the TPS6123x datasheet. They help to stabilize the regulator. The inductor and the output capacitor C3 serve as energy storage during conversion. Both the EN, HYS and the power good, PG, pins can be left floating or unconnected if not used. At moderate or heavy load currents, the converter would operate at a 2 MHz frequency pulse width modulation (PWM). At light load current, it reduces the switching frequency and operates with pulse frequency modulation (PFM).

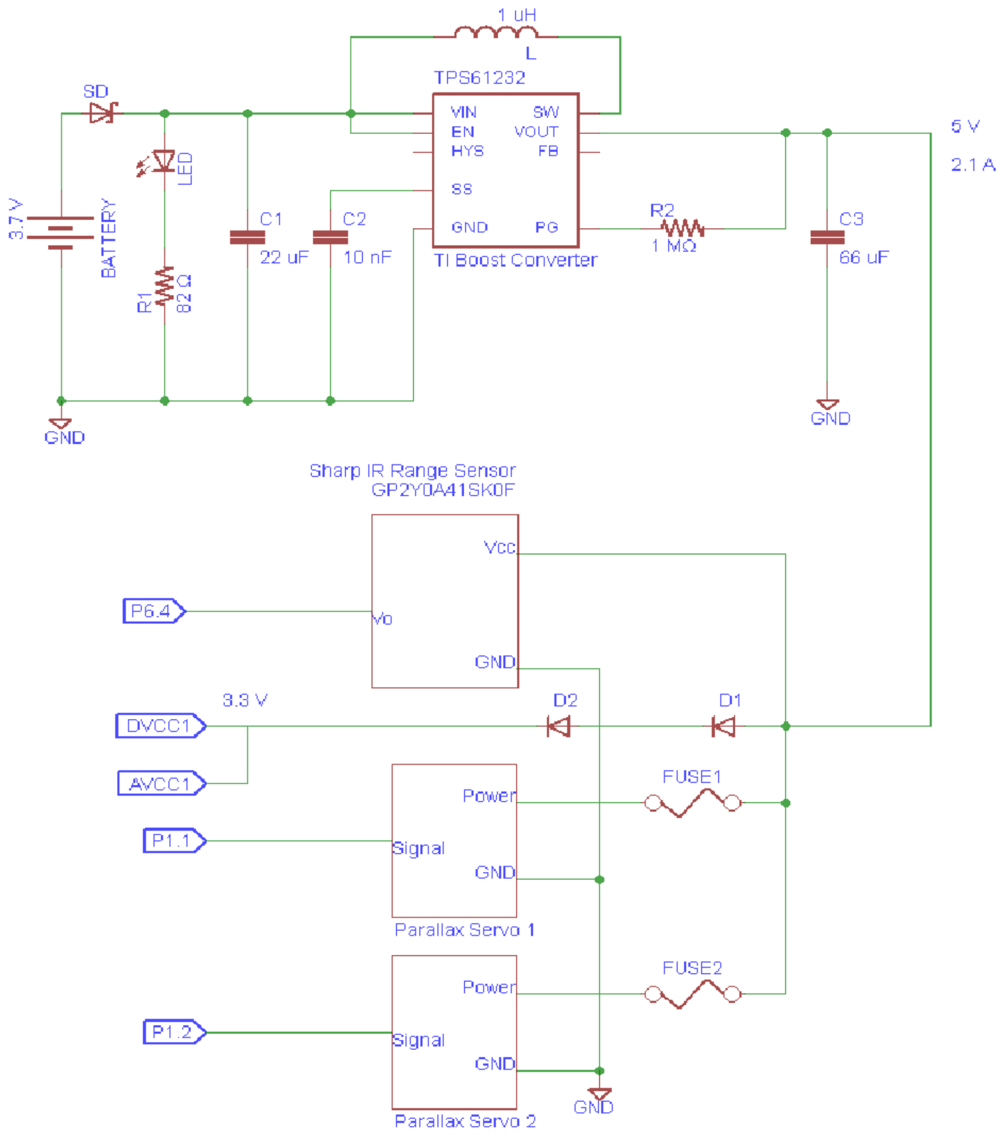


Figure 3.6: Schematic for Boost Converter

Buck/Boost Regulator Circuit Design

If our team decides to use a buck/boost converter to regulate voltages, then a TI TPS63061 converter will be selected. Out of all the other TI buck/boost converters, it is one of the few that is able to regulate a fixed voltage of 5 V. Besides, its 93 percent efficiency is high. It can also accept input voltage range from 2.5 V to 12 V, making it ideal for low voltage supply. Nevertheless, it’s not advisable to have a converter’s input voltage at the exact minimum and maximum of the range since the input might deviate from the values. Going off the input range would result in a damaged converter or one that doesn’t regulate voltage at all.

Unlike the TPS61232 boost converter, the TPS63061 has additional pins with special functions. The PS pin is used to enable/disable power save mode. A 1 is disabled and a 0 means enabled. During power save mode, the switching frequency and quiescent current is reduced to maintain high efficiency. Disabling the PS would set the switching frequency at a fixed rate. Connecting a clock signal at the PSY/SYNC pin would force the converter to synchronize to the clock’s frequency. To enable the EN pin, set it to 1. Otherwise, set it to 0. In many applications, the pin is tied to the supply voltage, which is on high. Hence, the pin is always enabled. To shut down the device, the EN can be connected to the ground. The battery and the load are disconnected during shutdown. The power good or PG indicates whether the output voltage is regulated properly. For the PG pin, setting to 1 means good, 0 means failure.

The converter has additional features including overvoltage protection, over temperature protection, short circuit protection, under voltage lockout, and power save mode. Once the temperature goes beyond a threshold, the IC stops its operation. As the temperate decreased below the threshold, the device starts operating. The under voltage lockout functions by automatically starting the device only when the supply voltage on VIN is above a certain under voltage lockout threshold. If the supply voltage goes below the threshold, then the IC automatically enters shutdown mode. The overvoltage protection internally monitors the output voltage so that it doesn’t exceed critical values. There is no timer in the IC. As a result, the output voltage overshoot and current inrush occur at startup but the device keeps the current and overshoot at minimum. When the output voltage does not rise above 1.2 V the IC would assume a short circuit at the output and protect itself by keeping the current limit low, typically under 2 A. The efficiency rises as the output current increases. The output current depends on the input current from the battery. As a result, the battery will be selected to have the current rating as high as possible. Figure 3.7 shows a circuit design with the TPS63061.

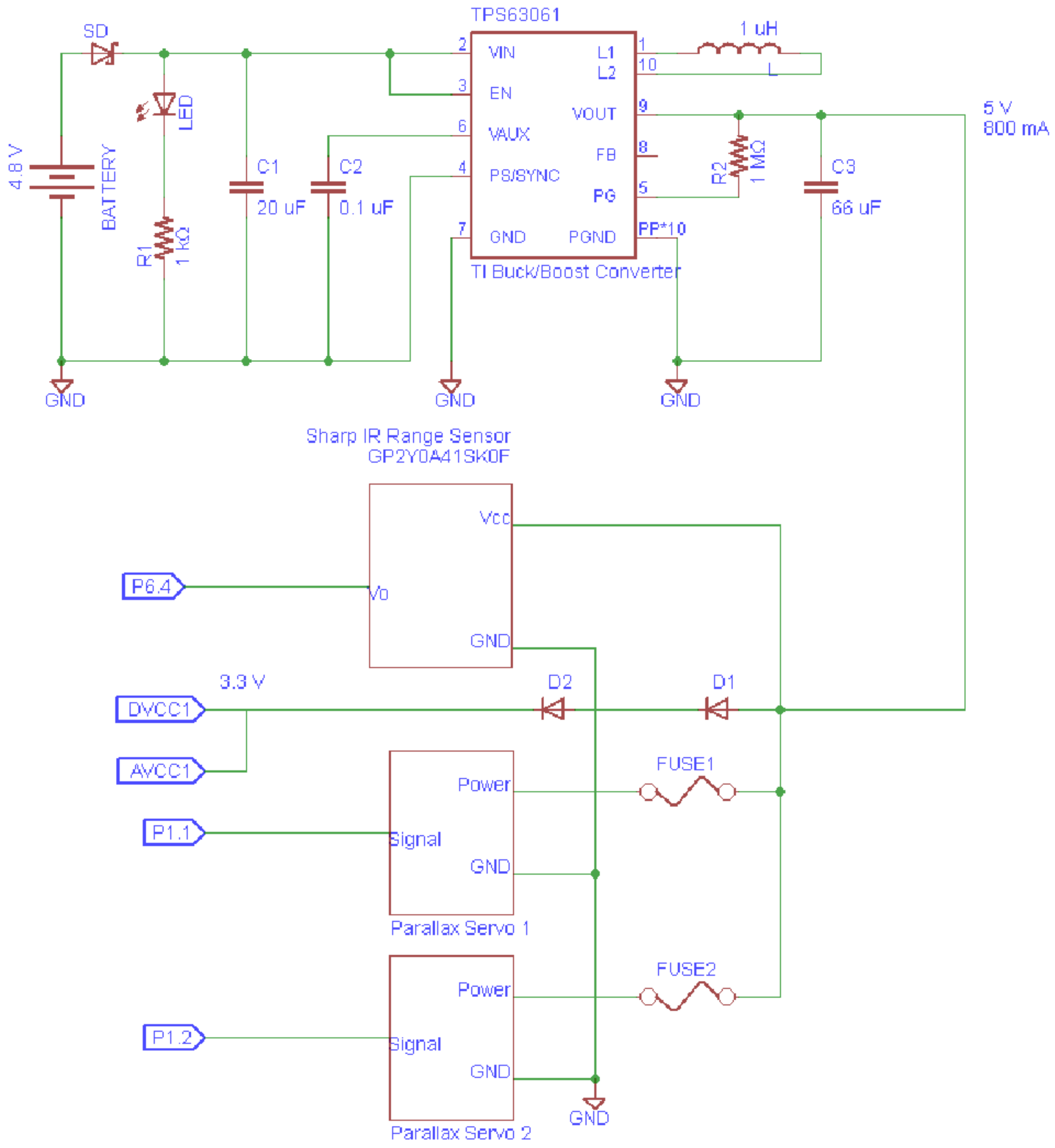


Figure 3.7: Buck/boost converter circuit with motors at output

An additional design for the buck/boost converter circuit would be separating the microcontroller and sensor subsystem from the motors and placing the motors at the input of the voltage regulator as opposed to the output. This configuration could be more advantageous because if the voltage regulator fails, the motors will not be affected. Figure 3.8 shows this configuration.

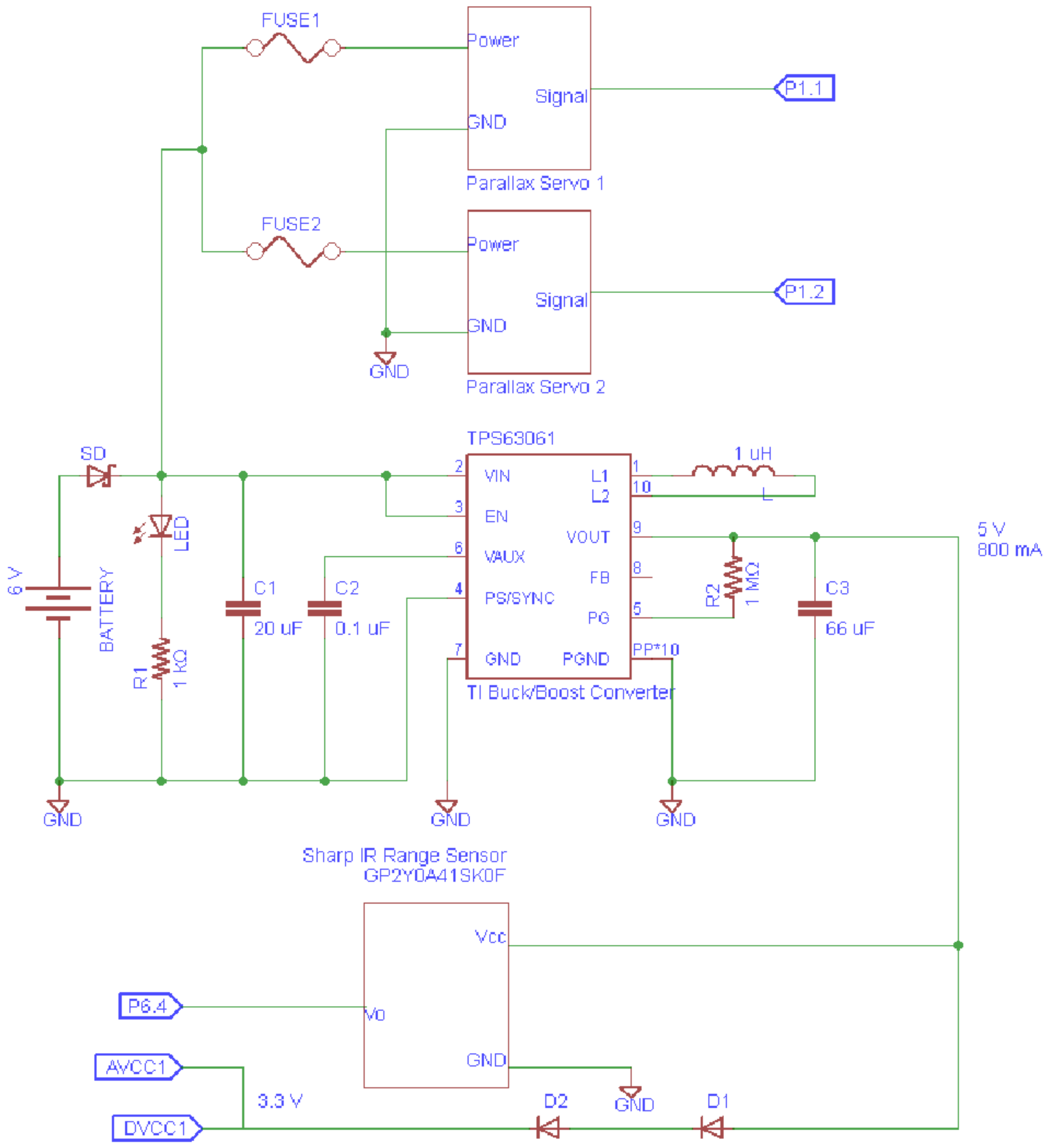


Figure 5.8: Buck/boost converter with motors at input

Linear Regulator Circuit Design

This circuit is better than the circuits that use switching regulators since it has a smaller number of components. Since the regulator can only output a maximum of 500 mA and its efficiency is low, it could be used to power the low-power devices such as the sensor and microcontroller. For the components that require higher current such as motors, they will be directly connected to the battery instead of the regulator's output. The LM2937-5 should be used because it has a fixed output voltage of 5 V.

Unlike the TPS61232 and TPS63061 converters, the LM2937 has reverse battery protection. As a result, a Schottky diode should be connected to the fuses to protect the motors. The regulator's reverse battery protection circuit automatically protects the sensor and microcontroller. Therefore, a Schottky diode is not needed at the regulator's input. Though the typical minimum dropout voltage is 0.5 V, the input voltage should be at least 2 V higher than the output voltage for optimal performance. In other words, the input voltage is required to be at least 5.5 V but should be 7 V or higher. Because it's harder to find a 5.5 V than a 6 V NiMH or Li-Ion battery, a 6 V battery would be used with the LM2937 regulator. The regulator's quiescent current is typically 10 mA if the regulator is under full load and the input and output voltage difference is greater than 3 V.

The LM2937 has additional features including thermal shutdown, short circuit current limit, and overvoltage shutdown. The thermal shutdown circuitry is not intended to replace the heat sink. Running the IC at thermal shutdown is not advisable because the device's reliability may be degraded as the junction temperature rises above the allowed absolute maximum junction temperature rating. In cases the output is shorted to ground or the load impedance is extremely low, the device would limit the current. If the LM2937 operates continuously at the current limit, then the IC would transition into thermal shutdown mode. Since our project would not use any power supply that exceeds 26 V we have no need to be concerned about the overvoltage shutdown. The LM2937 lacks the under voltage lockout and enable functions. The output only tracks the input voltage until the input rises above 6 V where the device remains in linear operation. Figure 3.9 shows circuitry using a LM2937 LDO (low-dropout) linear regulator.

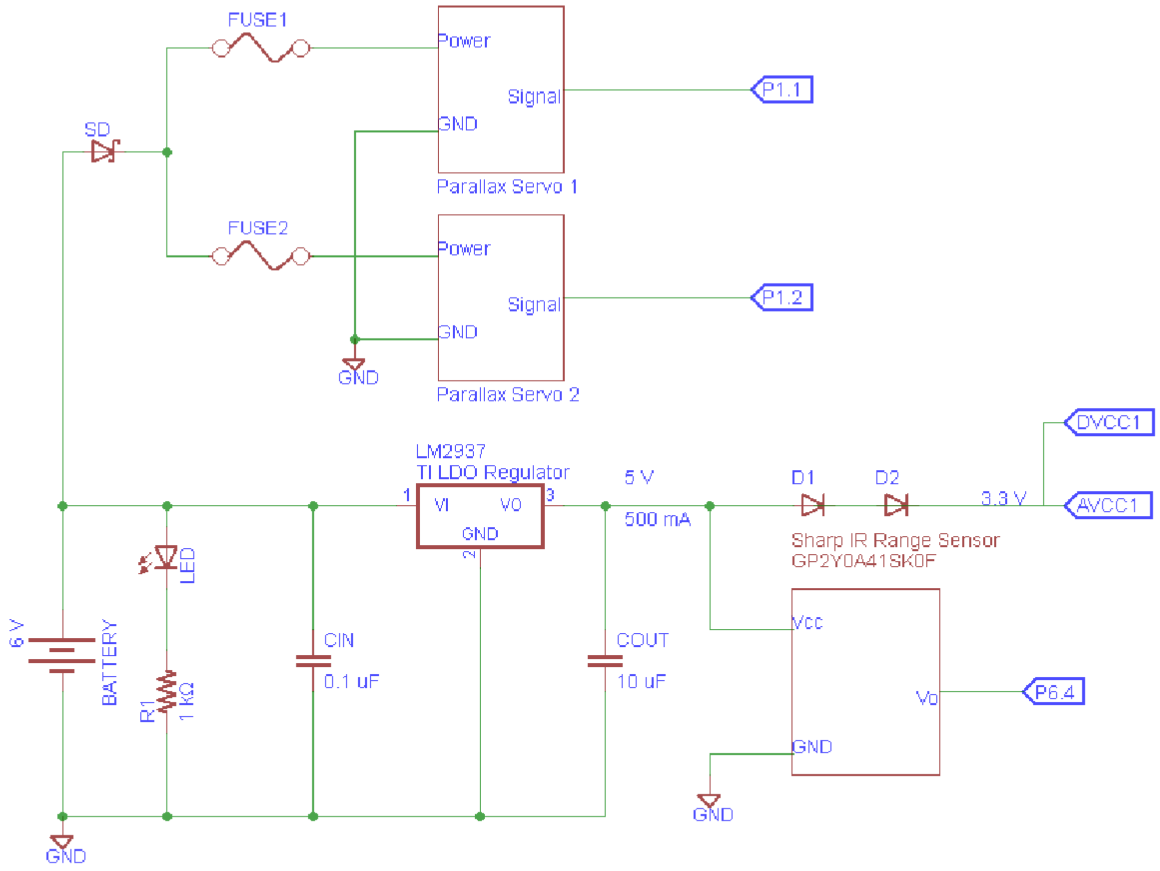


Figure 3.9: Schematic for LDO Regulator Circuit

Voltage Regulator Selection

The choice for the voltage regulator was between the LM317 and the LM3940. They are both linear regulators because they are simpler to use than switching regulators. Initially the LM317 regulator was going to be used because of its simplicity and familiarity using them before in past labs. All it needs is one resistor to set the current output. Then the regulator LM3940 was discovered by the group. The LM3940 is even more simple to set up than the LM317 and it was perfect for KittyBot. The LM3940 takes in 4.5-5.5 volts and steps it down to 3.3 volts. The max voltage is important to note because the batteries measured supply voltage was around 5.3 volts. No resistors were required to adjust the current, the LM3940 outputs 1 amp which is more than enough for the MSP430 microcontroller. Capacitors were placed in parallel to ground on the input and output to reduce noise in the signal as seen in figure 6.1.4.2 below. In the end the LM3940 was chosen because its input takes the same voltage as the motors and ultrasonic sensor as well as its output is the perfect amount for our microcontroller.

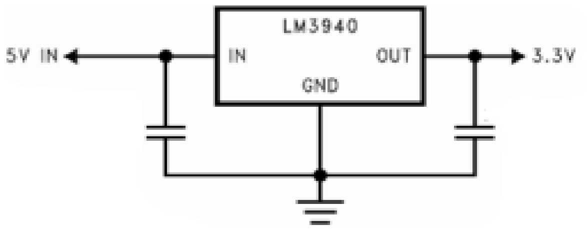


Figure 3.10 (From www.ti.com)

3.3.4 Gyroscope

An exciting strategic component we researched is the gyroscope. A gyroscope is a small electronic device, many are no larger than a quarter, that measures rotational motion. **Figure 3.3.14** display what this type of module looks like. Gyroscopes can measure angles and angular velocity. Angular velocity is measured in degrees per second or revolutions per second. Angular velocity is the measurement of the speed of rotation. The process of gyroscopes measurements is when the device is rotated, a tiny resonating object inside the gyroscope is shifted as the angular velocity changes. The shift is converted into a low-current electrical signal that is amplified and read

Figure 3.3.14: LPY503 Gyroscope

(From learn.sparkfun.com)

This can help our project because the gyroscope can detect changes in orientation. The changes in direction can be measured from a set balanced position and corrections can be sent to the motors. In 3D space, there are three axes X, Y, and Z (**Figure 3.3.15**). Objects can rotate about any of the three.

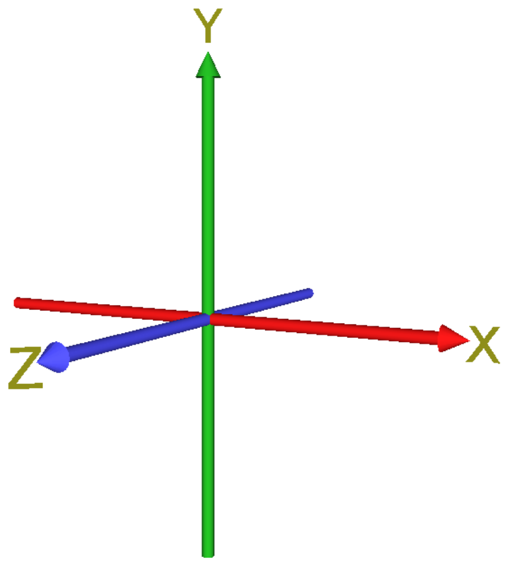


Figure 3.3.15: XYZ Axes

Gyroscopes come in varieties that either measure rotation around a single axis, two axes, or all three. The price difference on these varieties is minuscule these days, but we may only need gyroscopic detection in one axis. **Figure 3.3.14** displays a 3D representation of sphere rotation.

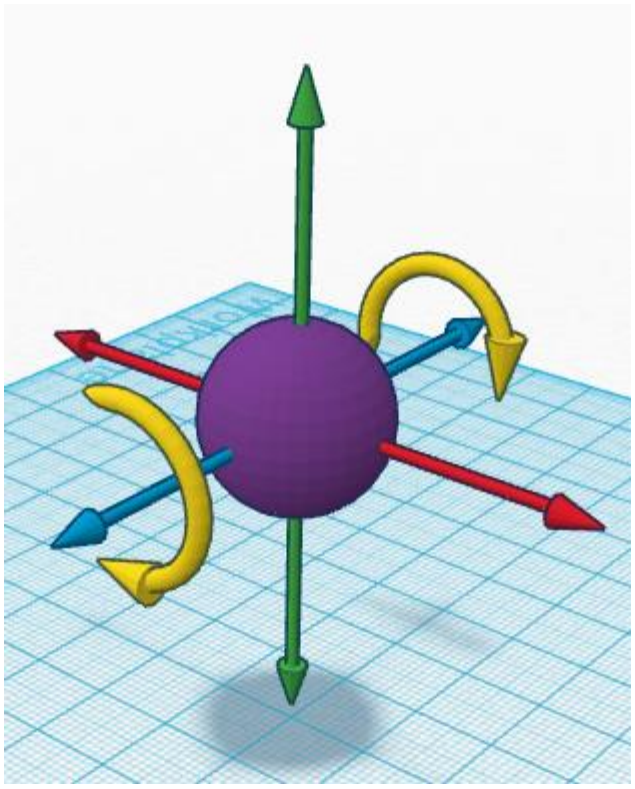


Figure 3.3.16: 3D Representation of Sphere Rotation

Forward motion is the primary focus. **Figure 3.3.16** uses the same colored axes as **Figure 3.3.15**. The yellow curved arrows indicate rotation about the blue Z-axis. This rotation will cause the sphere to roll forward. The gyroscope, in this cause, would detect rotation in the Z-axis. Most standard gyroscopes are not meant for picking up very fast spinning objects. Luckily KittyBot is an indoor toy so its rotations shouldn't be too fast for a gyroscope to measure. The forward linear velocity or acceleration won't affect the gyroscope either, as it only picks up and measures angular velocity.

Gyroscopes connect through power and through a communication interface. The communication interface can either be analog or digital. Digital communication can be through Serial Peripheral Interface or Inter-Integrated Circuit. Serial Peripheral Interface, or SPI, is a type of interface bus used to send data between a microcontroller and peripherals. SPI works synchronously as opposed to asynchronously.

Communication

A standard serial port with RX and TX (think receiver and transmitter) lines, typically works asynchronously. This means that the rate at which data is sent and received is not controlled, and that lack of control comes from the two side not running at the exact same clock rate. Computers have everything synchronized to a single clock, but when you try to communicate between two different computer systems, like a microcontroller and its peripherals, the clock rates may be different. To make asynchronous serial communications work extra bits are added to the end of the data-stream. A start bit at the beginning and a stop bit at the end help to isolate the desired bits allowing the receiving system to sync up with the data properly. The two separate systems must be set to the same transmission speed beforehand for this to work properly. **Figure 3.3.17** aids in displaying this method of serial communication.

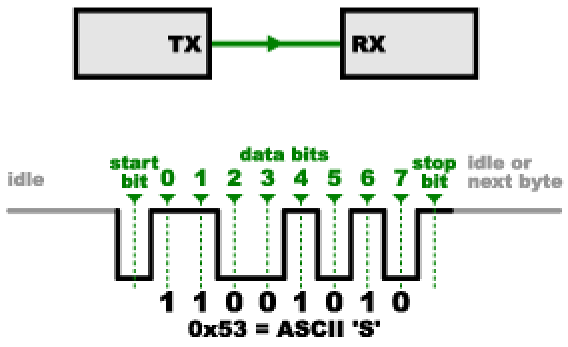


Figure 3.3.17: Asynchronous Serial Communication

(From learn.sparkfun.com)

Asynchronous communications work but are tricky due to the number of complications. The start and stop bits have to be sorted out in order to get the correct data, and the transmission speed have to be the same, if they are not than the data sent will be wrong.

SPI's synchronous communication works differently. The data bus for a synchronous serial uses separate wires for data sent and a clock that keeps the communicators synced. The clock is sent out as an oscillating signal. This signal tells the receiver when to sample bits from the stream of data. The receiver picks up on the rising or falling edges of the clock signal. A rising edge is a shift from low to high and a falling edge is from high to low. Whichever edge that is set to be the "triggering" edge for sampling will tell the receiver that sample at that moment. Below is **Figure 3.3.18** displaying this type of communication.

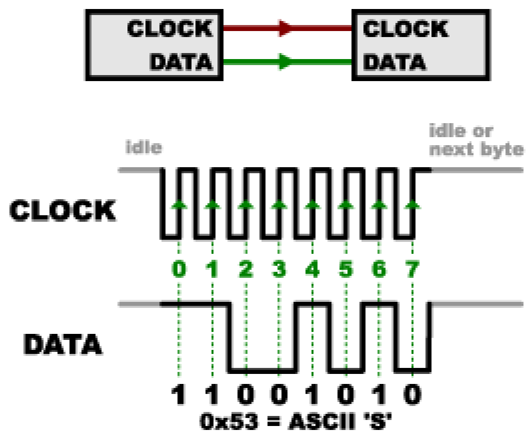


Figure 3.3.18: Synchronous Serial Communication

With SPI, the side that generates the clock is called the "master" and the other is called the "slave". In the case of our embedded systems, the master will be the microcontroller and the slave, peripherals such as a gyroscope. The microcontroller is always controlling when data is sent, and sends commands to the gyroscope so it will send its angular velocity data back to the microcontroller for processing.

SPI offers these advantages and disadvantages.

SPI Advantages:

- Supports multiple slave receivers
- Receiving hardware can be as simple as a shift register
- Faster than asynchronous serial

SPI Disadvantages:

- Requires more wires than other communication interfaces
- Master must control all communications
- Communications must be defined in advance
- Usually requires separate SS lines attached to each slave.

Our other digital interface is the Inter-Integrated Circuit, or I2C. I2C allows multiple slave systems to communicate with one or more master systems. I2C has an advantage over SPI in that it requires less connections. SPI needs four lines to connect a single master to a single slave, one for the clock, one for sending data from the master to the slave called “Master Out/Slave In” (MOSI), one for sending data from the slave to the master called “Master In/Slave Out” (MISO), and the “Slave Select” line which acts as a sort of “wakeup” command sent from master to slave readying the slave to send or receive data. Besides the pins taken up on the master unit by these lines, additional slaves require an additional chip select I/O pin. The SPI interface has a tendency of filling up several pins very quickly. This is problematic if you have only one master and multiple slaves. With KittyBot, our group wants to keep the internal space as uncluttered as possible since it is limited and must all fit within a relatively small form factor. That is why we would want to keep the number of microcontrollers to just one. Budget is also a concern if we need to purchase additional microcontrollers. The gyroscope is not the only slave our microcontroller would have as we are considering other strategic components to add as well, so pin space on a single microcontroller is limited.

I2C’s required number of lines is merely two and that is for multiple slave devices, up to 1008 to be exact. The two signals sent through I2C are SCL and SDA. SCL acts as the clock signal while SDA is the data signal. Similar to the principles discussed with SPI, I2C’s master device sends the clock signal which controls the sending and receiving of data. With I2C however, the slave has an extra ability. Let’s say the rising edge is the signal from the master to send/receive data and one is approaching. That means the clock is currently low and an oscillation up to high is coming. The slave has the option to force the signal to remain low in order to delay the master if the slave is not yet ready to send/receive. This ability is called “clock stretching”. The I2C bus operates on what is called an “open drain”. Remember that the bus connection can be between a single master and multiple slaves. All the slaves have this clock stretching functionality. The idea behind the open drain is that any slave can drive the clock signal low if it needs more time, but none can drive it high if they are ready. This makes it so that the transmissions only go through when all the devices in question are ready, preventing potential damage to the transmission. A pull-up resistor on the signal lines are used to restore the signal back up to high if no device is forcing a low signal. **Figure 3.3.19** displays a representation of this.

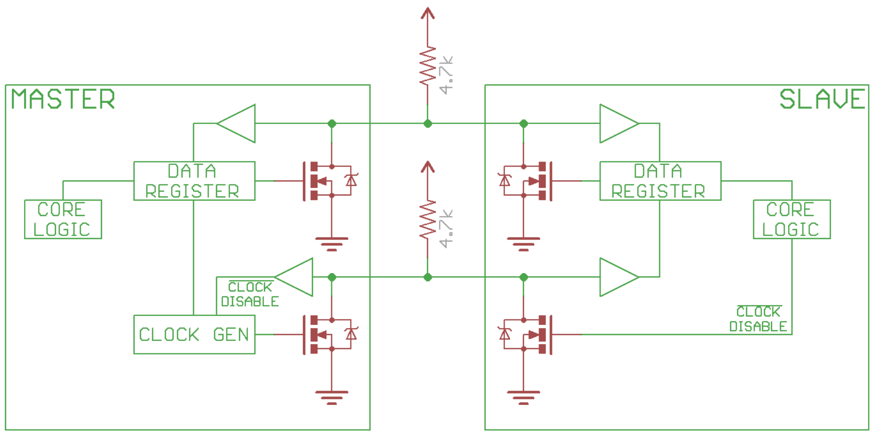


Figure 3.3.19 Generic Master/Slave Connection with Pull-up Resistors

(From learn.sparkfun.com)

Both SPI and I2C are intended for use over a short distance. Since all electronics will be confined within the space of KittyBot’s chassis, distance should be an issue. These two also have limitations on their sampling rates, with SPI reaching higher rate than I2C. This could potentially lower the accuracy of the angular velocity readings.

Those were the digital communication options. The alternative to digital is analog communication. The gyroscope can register rotational velocity by raising and lowering voltage between ground and the supply voltage. Analog gyroscopes usually run cheaper than digital gyroscopes and can even be more accurate. The accuracy depends on how the analog signals are read. Analog to Digital Converters (ADC) need to be used to transfer those analog signals to digital ones that a microcontroller can process. Voltage is the type of analog signal detectable by microcontrollers. Only certain pins on a microcontroller are capable of this as well. **Figure 3.3.20** shows a comprehensive guide by Texas Instruments to TI MSP430.

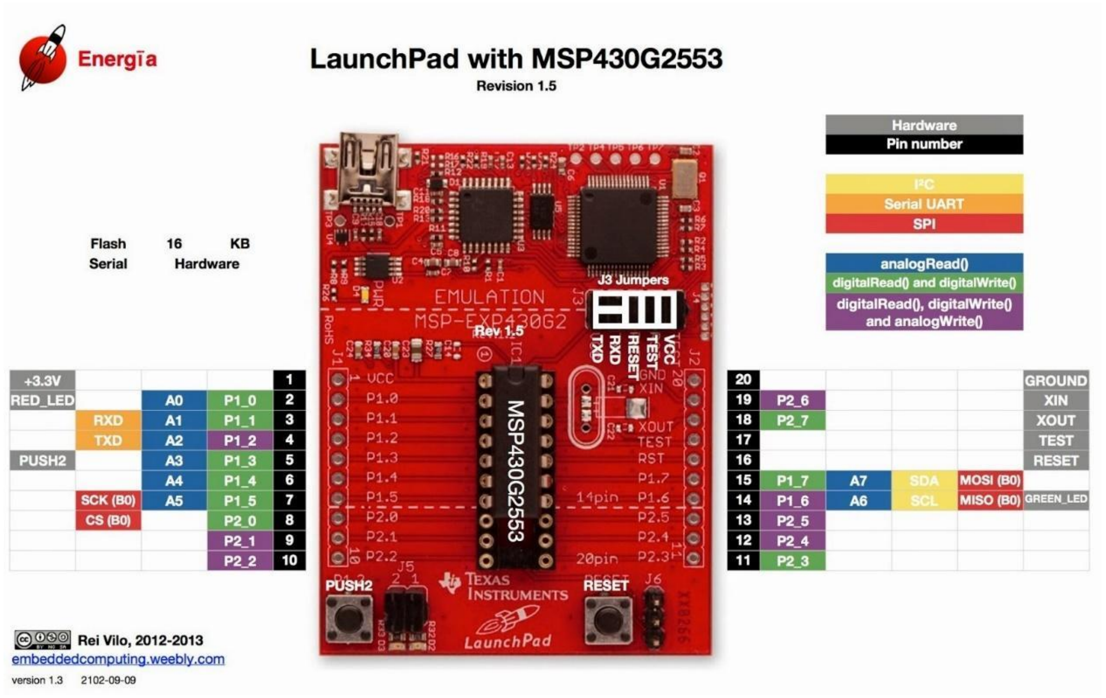


Figure 3.3.20 TI MSP430 Launchpad

(By Texas Instruments)

The TI MSP430 is our groups microcontroller of choice. This microcontroller has 8 pins, P1_0 through P1_7, that are capable of receiving analog voltage signals. These signals can then be converted to digital. The ADC of the microcontroller handles the actually conversion. TI MSP430's ADC is 10-bit meaning it can detect up to 1024, which is 2^{10}, discrete analog values. The actual conversion is done by the analog voltage to be converted is used to charge an internal capacitor that is then discharged across and internal resistor. The time of that discharge is measured by the microcontroller counting the number of clock cycles that pass between the time the capacitor began discharging to when it stopped. The number of clock cycles is then returned to the microcontroller as the new digital value. The maximum value of a 10-bit ADC is 1023 because it can have 1024 different values ranging from 0 to 1023. The maximum value digital value has a ratiometric relationship with the overall system voltage, or V_{CC}. This means that the digital value of any analog value sent to an ADC-capable pin is a ratio of 1023 and the V_{CC}. Assuming a V_{CC} of 5 V and a measured voltage of 2.5 V, the digital value can be described by *x* in the following equation.

$$\frac{1023}{5.00V} = \frac{x}{2.5V}$$

Solving for *x* we would get the following.

$$\frac{1023}{5.00V} * 2.5V = x$$

$$\frac{1023}{2V} = x$$

$$x = 511.5$$

The digital value would end up being 511.5. Notice that the measured voltage value of the ADC was 2.5 V, which is half of 5 V. The converted value of 511.5 is also half of 1023. In order for the microcontroller to pick up these values it would to be programmed to do so. It would first need to define the pin it is receiving input from. Let's assume pin P1_0 is the receiving an input voltage for this example.

```
pinMode(P1_0, INPUT);
```

Tell the microcontroller to convert the input from analog to digital with the analogRead() command into in integer, x.

```
int x = analogRead(P1_0); //Reads the analog value on pin P1_0 into x
```

This is how the analog voltage values sent from a gyroscope can be processed by our microcontroller.

Power

Gyroscopes are typically low power devices, so powering the device is not a major concern. The levels of current to operate them fall in the milliAmp or even the microAmp ranges. Digital gyroscopes can operate at the supply voltage or have their own set logic levels of voltages. The digital gyros need to be configured more carefully because they need set logically states to operate. Digital signals are binary, 0 or 1, ON or OFF. The digital gyroscopes may seem more finicky, but they can also have a low power and sleep mode. This can converse more power in the long run versus an analog gyroscope. Since KittyBot is a battery-powered unit, consuming less power will allow for longer operation times.

Moving on, the specifications of the gyroscope are key. A few different specifications to look at are the gyroscopes range, sensitivity, and bias. The range, or full-scale range, of a gyroscope is simply the maximum angular velocity a gyro can read. This does not need to be too drastically high for this project as KittyBot will not be reaching very high speeds.
The sensitivity of a gyroscope is how much the voltage changes for a given angular velocity. Sensitivity is measured in millivolts per degree per second (mV/°/s). A gyroscope, just like any sensor, contains some degree of error. This is called the bias of said sensor. Gyroscope bias can be seen when the gyroscope is still. Instead of being exactly 0 degrees, the gyroscope will always read a slight non-zero value in the output. This bias drift or bias instability can be caused by a few different factors. Temperature can be a major factor. This is alleviated by most gyroscopes by having a built in temperature sensor. This data can be read and used to correct any temperature dependent changes. Calibrating a gyroscope correctly is the best way to reduce error. This can be done by keeping the gyro still and zeroing all the values and readings in the code on the microcontroller.

3.4 Possible Architectures and Related Diagrams

In this section the various architectural designs for Kitty-Bot are discussed. The resulting research and conclusions based off of said research are also detailed.

3.4.1 Design Choice: Spherical vs. Dual Motor

When Designing the KittyBot we quickly realized considering several architectural designs would be a crucial step in efficiently completing the electrical and software model. When mentally visualizing our design in hopes of arriving at an intelligent solution we had to consider all the advantages and disadvantages of any system and especially the consequences they would inflict on the hardware power supply and software. For example, one of our design constraints stated the robot should be durable enough to withstand potential damage from an animal. With this in mind any potential structural design that included many moving parts and could potentially have damaged hardware would have a negative weight towards its final consideration. We need a structural design that is not vulnerable or fragile, being that the cat would quickly devastate the structural integrity of our unit and its hardware if it was so. Structural integrity is especially hard to maintain in terms of the sensors. A sensor can very easily be disconnected or damaged so any structural design had to take this into consideration. Another requirement was that our robot should be maneuverable. That being said, the challenge lies in making the unit maneuverable without having allot of moving parts. When considering how the structural design would affect the hardware power supply and software, we found many interesting and unique designs that were quickly disqualified because of their complex attachments and multiple moving parts; These we found out would be antagonistic towards our hardware simplicity. Also, from a purely practical point of view, avoiding moving parts in particular was a hurdle to jump when trying to ensure that the robot would erect itself upright if turned over. In short, we quickly realized that any structural consideration would undoubtedly impact the electrical design in either a positive or adverse way.

When we considered all of the previous design constraints in tandem with potential structural designs we arrived at two final candidates, the Spherical Design and the Dual Motor Controlled Wheel Navigation Design. We kept our printed circuit board and Power Supply design at premium thoughtfulness when deciding which route to choose. Both candidates satisfied our design limitations within a reasonable margin. Both considerations, however, contained both virtues and faults. Our challenge would now become deciding what design consequences we were comfortable with enough to confront. One of the candidates was the spherical design. This design would consist a spherical outer shell encapsulating the inner hardware power supply and sensors. The hardware would contain sensors on the inner part of the sphere that would create forces to actually spin the structure. This architecture, we realized, would be far more difficult to design in a structurally sound manner. This due to the fact that everything would be rolling. This would mean that the microcontrollers battery pack, fame and motors would somehow need to be stabilized. However, the final product would present significant increase in elegance to our solution. The other candidate was the dual motor controlled wheel navigation design. This design implements a hardware simplicity driven structural architecture. The structural architecture facilitates ease of wiring between the power source, microcontroller and sensors.

In one of our group meetings we came up with the following brainstorming pros and cons. The brainstorm resulted in **Figure 3.4.1**. This tree of pros and cons was a great way to start considering which of the two designs we actually wanted to invest more intellectual resources in. We would later, as demonstrate in the figures following this paragraph, 3D models our designs to further consider them and eventually even proto type both of them in a physical sense. But we started out with this very simple brainstorm of pros and cons to each design. The Dual motor controlled wheel navigation cylindrical design we reasoned would have a relatively simple architectural design due
to the physical proximity of all the parts. The sensor would be on a stable platform in this structural architecture as opposed to the spherical design where we would have to design for a spinning structure where sensors would be fundamentally less stable. Also with the dual wheel which we tested on a Bo-Bot we found extremely precise maneuvering capabilities were within reach. The disadvantages we reasoned started with the visually appealing and non-elegant architectural structure upon which all of our electronics would rest on. This, as opposed to the rolling sphere which most certainly had a “Cool” factor to it. Another disadvantage we reasoned was the relative openness of the hardware. With the cylindrical design, it would be harder to keep the hardware from being dislodged during play with the cat. This, as opposed the Spherical design which would by nature encapsulate all of the hardware in a sphere. With the sphere there was the advantages of it being “Cool”, having the hardware enclosed and also visually appealing. With all of the previous in mind we would eventually need the help of 3D models to make our final decision. which would eventually be to take the risk and implement our spherical design. The 3D model would help us consider the electrical and power implications.

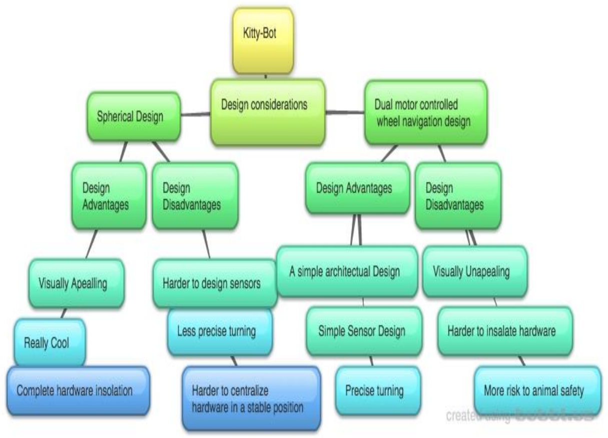


Figure 3.4.1 Pros and Cons Flowchart of Two Designs

3.4.2 PCB, Sensor, and Power of Spherical Design

This pros and cons brains storm was of course a great tool to choosing our final design. Our decision, by unanimous vote would be the Spherical Design. We knew that it would be far more challenging than the dual motor controlled wheel navigation design, but we were really excited about how cool a rolling sphere would be. We decided to disregard our design fears of tackling a more complex system and trust our University of Central Florida Engineering training to hopefully get us through the problem solving difficulties. However, we needed more to truly map out our design giving that our academic careers and countless hours of intellectual resources that would be spent developing one what we decided would be the Spherical Design. A couple of senior design meetings later we arrived at the conclusion that we needed 3D models of our designs to further investigate the electrical and power consequences of our conclusions. We did not need to make them completely perfect but rather close enough to leverage our imaginations. This way we could more closely consider the printed circuit board implications of our chosen path.

3.4.3 Dual Hemisphere Structure

The core Electrical design advantage to this candidate was its spherical nature. That being said, the core structural design disadvantage to this candidate was its spherical nature.

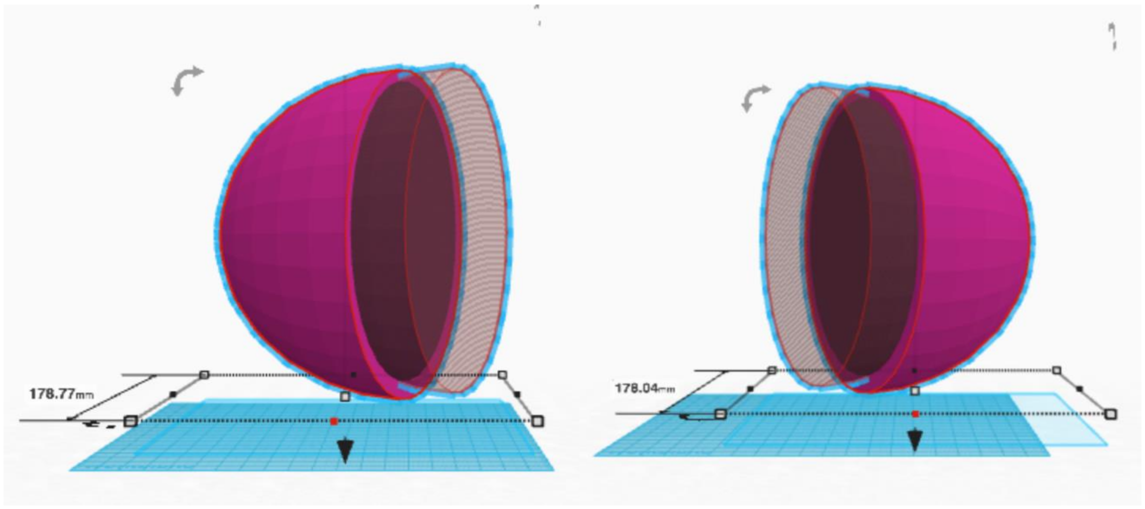


Figure 3.4.2 3D Model of Right and Left Hemisphere

We imagined a structural architecture composed of a hollowed out sphere. We 3D modeled this idea in **Figure 3.4.2**. This idea was birthed as result of all the structural shortcomings of the our first prototyped dual motored control navigation design. We imagined a hollowed out sphere that would have two hemispheres, a left hemisphere and a right hemisphere. The two hemispheres would be manually detachable allowing us to access the hardware inside if necessary. We imagined this sphere being approximately one hundred and seventy-eight millimeters in diameter. This would give us enough room to insert electronic hardware, disassemble parts and still be within the limitations of our system requirements. The next question would then become, how do we put hardware in a sphere and make it roll?

3.4.4 The Containers' Battery Pack and PCB Design

In order to have hardware in the sphere we would need a frame to place it on. This frame would have to be small enough to fit in the sphere and large enough to hold all of the electronics. We decide to call this frame our "container." On top of this container we also needed to place our electronics and microcontrollers that would eventually become our printed circuit board. And, of course, we would need a power supply to power the entirety of the system. We realized that we would have another layer of complexity in trying to make certain things don't fall apart. This due to the fact that everything would be rolling. This would mean that the microcontrollers battery pack, fame and motors would somehow need to be stabilized. However, the final product would present significant increase in elegance to our solution so the levels of difficulty would be worth it.

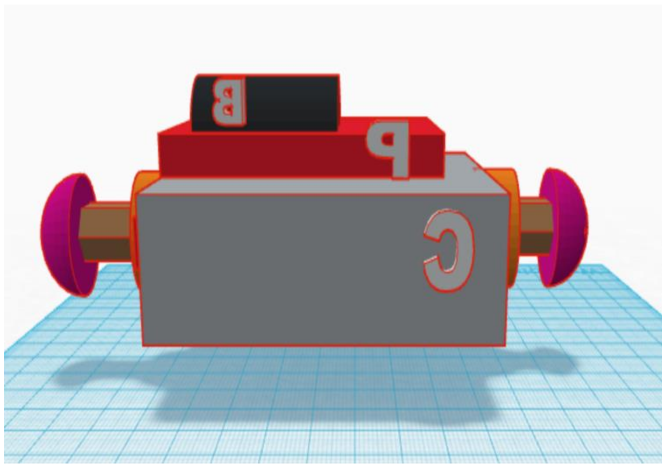


Figure 3.4.3 3D Model of Center Container

Figure 3.4.3 is the 3D model representation of what we imagined would be a good design for the inner electrical and power devices. Notice the “C”, “P” and “B” on the AutoCAD model these were placed to assist us as well as the reader. The “C” is the container. This is what holds all of the electronics. The “P” is the printed circuit board. This is what would be powered by the battery and control all of the sensors. The “B” is the battery pack, this, of course, powers the entire system. We reasoned this was a good structural design. It could be fit inside of the sphere and operate in an elegant fashion. This was the basic blueprint of our design. Functionally, however, we still needed to spin the entire unit in order for the ball to roll. During our group meetings we had many suggestions as to how to actually accomplish this. One of the suggestions was very practical yet simultaneously silly. A group member suggested that we put wheels at the bottom of the container and turn the entire platform into what could essentially be described as an RC car in a sphere. This would turn the entire sphere and propel our system. This idea, though practical and potentially easy to implement had many drawbacks. For one we didn’t like the lack of preciseness and instability. We would have no way of precisely knowing how much we would turn. Furthermore, the inner structure would need an entirely separate design to withstand the impact. It would be rolling around inside the sphere in a completely unstable fashion not fastened to anything at all. Needless to say, we quickly dismissed this suggestion. We brainstormed and conversed amongst each other contemplating different solutions and finally decided we should simply have the motors spin the sphere itself.

3.4.5 3D Modeling to consider PCB Implications

When designing and thinking about our printed circuit boards, power supply and hardware in the spherical model we needed to give careful consideration to the structural designs effect on the printed circuit boards. Circuits, the most recent sensors, and code are vital parts of a hardware venture and this spherical design would undoubtedly affect how we designed these. Not considering how the physical surface of your device effects the electronics power supply and Printed circuit board design can bring about reliability problems and unwavering quality issues.

For example, we took into consideration what the trace might be on our printed circuit board. With an estimated current of twenty-five milliamps at an ambient temperature of seventy-eight degrees Fahrenheit we might expect perhaps a trace length of five inches and a required trace width of 0.000526 mil. With the spherical architecture we realized we could keep all of the components centralized limiting the trace of the printed circuit board. That being said it would still be difficult to design in a structurally sound manner. However, the final product would present significant increase in elegance to our solution. We plan on using length width and thickness of trace to control resistance. The centralized printed circuit board design of our spherical architecture will make these calculations easier to handle. The electrical team will eventually have to make calculations for the trace with beige that we can’t change the physical properties of copper which creates resistance. Eventually we will want to aim for about a five-degree temperature rise. This being said the amount of space that the spherical design affords our PCB boards will help with making certain it doesn't overheat. Also, Circles or Loops in the PCB could be made small with

$$R = \frac{(resistivity * length)}{(thickness * width)}$$

this design. Little Loops have lower inductance and resistance. Putting circles over a ground plane further diminishes inductance. We could, if needed, lessen the voltage spikes in this way. Also, when the time comes if we were to use decoupling capacitors, we could place them close to ground and power. This would maximize decoupling efficiency by minimizing conductance. With this design we could also, if needed, keep Digital and Noisy traces away from analog traces. The centralized nature of all of our electronics and hardware would permit us to more elegantly design the PCB board when the time is right. We could very easily be mindful to course loud grounds from signs that should be quiet. Making our ground traces sufficiently large to carry currents that will flow would lower the impedance of the traces which would be ideal for us.

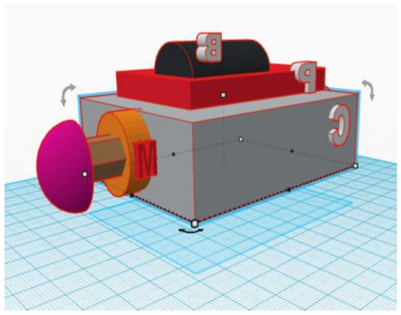


Figure 3.4.4 Angled View of Center Piece 3D Model

Consider the image displayed in **Figure 3.4.4**. Notice that now the Orange unit has a "M" on it. This is representative of the servo motor. Of course there would be another servo motor on the other side. The idea would be that the servo motor would spin the rod which would in turn be attached to either the right or the left hemisphere of the sphere which would then in turn rotate the entire unit. Now if we redirect our attention towards the pink elements in **Figure 3.4.5** and envision them hugging the inner wall of either hemisphere we can quickly realize how we plan on making the structure move.

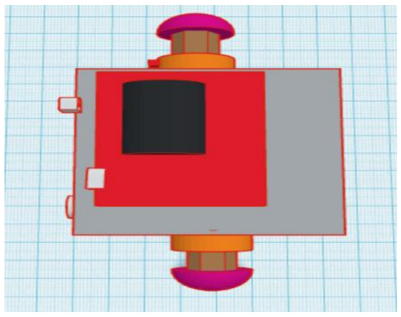


Figure 3.4.5 Top-Down View of Center Piece 3D Model

Of course the pink element would need to be fixed to the side of the sphere and the motors' casing would have to be stationary in relation to the rod and the right or left hemisphere in order for everything to turn. The force of the outermost pink element against the left and right hemisphere would create torque which would then create a force on the ground as a result of the weight of the unit and friction. The entire sphere, we reasoned, would then be forced to rotate as a result of this torque. We wrote our design thoughts down in the sketch in **Figure 3.4.6.**

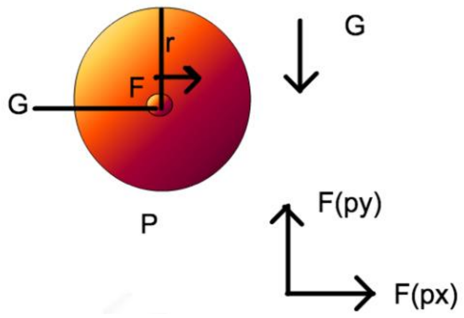


Figure 3.4.6

We considered the forces described in **Figure 3.4.6**. Here F is the force the pink unit in our simulation exerts on the sphere. The motor would be spinning the rod which would in turn be connected to either hemisphere of the sphere. This force would be the driving engine of our robot. r is the radius of our kitty-Bot. We were intending on making the Kitty-Bot about one hundred and seventy seven millimeters or seven inches so the radius would be about half of that. G is the center of mass of the sphere, g is gravities acceleration, which is, of course, 9.8 m/s2 and P is the point of contact of our kitty bot with the ground.FPx would be the x component of the force exerted on the our sphere by the ground, at point P. This would be a frictional force. FPy is the y-component of the force exerted on the our sphere by the ground, at point P. This would be a frictional force. Our group reasoned that the force exerted by the motors on the rod, would then create a force on either hemisphere of the sphere which would interact with FPx and FPy causing the Kitty-Bot to roll. Essentially, the outer pink ends of figure 1.10 would be spun by the motors, these pink ends would be bolted to either hemisphere of the sphere which would in turn cause a force “F” which would in turn cause torque. This would in turn create a force FPx and FPy against the ground “P” which would consequently rotate the sphere. This being said, when further developing the idea we ran into some logical problems. For one, what was going to keep our container stable?

If we observe **Figure 3.4.4** we see the container. In our design the motors would be attached but not fixed to the container which holds the printed circuit board and the battery pack. The motors would then turn the rods which would in turn be connected to the pink outer ligaments of **Figure 3.4.4** which would consequently turn each hemisphere. The key component to mention here however is that in our design we would Ideally want the container to remain relatively stable in conjunction with the rods being rotated by the servo motors. There are multiple reasons why we wanted this to be so. Firstly, a rotating container would mean rotating electronics, battery pack and printed circuit board. This would in turn mean we would have to fasten the electrical circuits including the sensors, printed circuit board and battery pack together so that none would fly off or become dislodged due to the forces caused by the case rotating.

When considering these rotating forces coupled with the cat’s interaction with the Kitty-Bot we arrived at many concerns. Secondly, if the container is designed in a way where it does not remain stable inside the sphere then this would undoubtedly cause a wobble in the sphere. This wobble would create undesirable results including a loss in precision and control in terms of steering the Kitty-bot. Thirdly, loss of stability in the container would mean our sensors would also be unstable. This would of course mean loss of accuracy in decision making. The nature of our Kitty-Bot involves the feline attacking the structural architecture as aggressively as possible. The sheer force of the impact of a Cats’ strike coupled with the rotating forces of a spinning container culminated in a plethora of concerns that resulted in us deciding to design a solution for this potential problem. We needed to find a way to keep the container stable while the motors turned the rods, this we would eventually decide to do with a counter weight.

3.4.6 Incorporating Useful parts of unused rapid prototype to our new design.

As mentioned before, the first candidate was the dual motor controlled wheel navigation design. With this design the hardware would have a simple to implement design. This simplicity driven structural architecture would make it easy to wire the power source, microcontroller and sensors together. Also, we could prototype it without too much 3D printing if necessary. **Figure 3.4.7** shows the essential parts of this structural architecture. The Cylinder, of course, would encapsulate the frame, power supply, central Intelligence system and sensors. The right, left, front wheel, Device Stabilizer and sensors would be ligaments exterior to the cylindrical exoskeleton. However, we thought Image **Figure 3.4.7** as more of a block diagram for the electronics and made decided to document and prototype the actual structural architecture in a 3D model shown further down in the document. Throughout the design process we actually prototyped this model and found many good qualities of it that we thought would be valuable assets to our final Spherical Design.

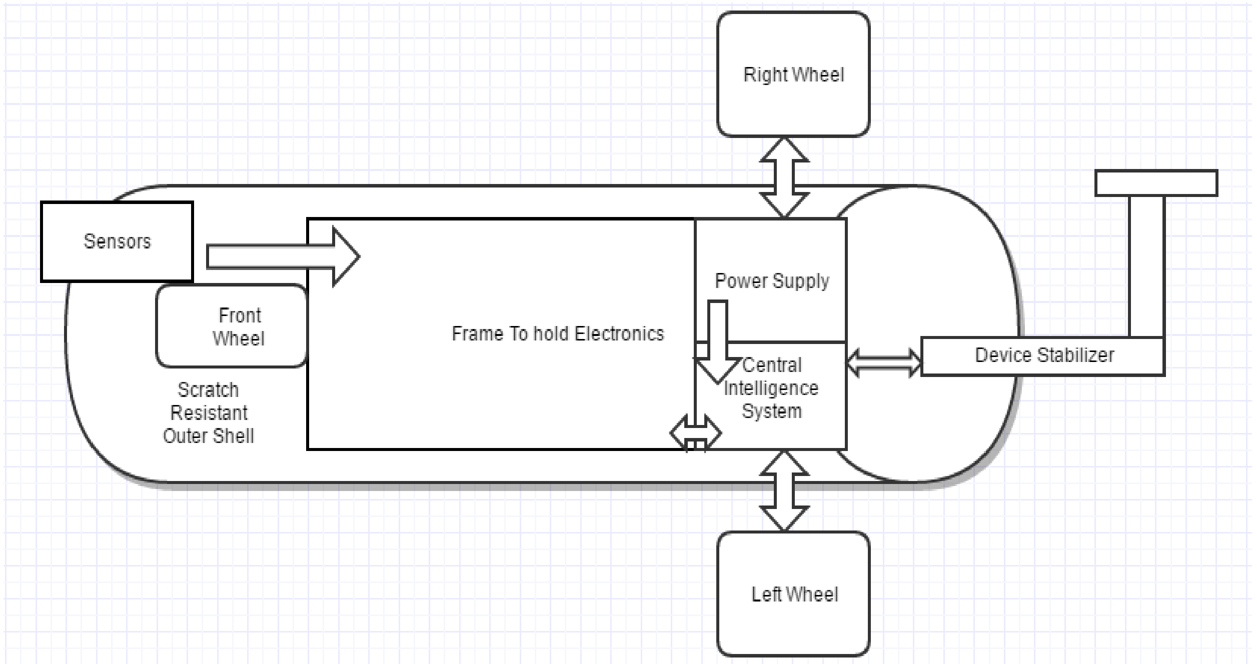

Right Wheel
Sensors
Power Supply
Front Wheel
Frame To hold Electronics
Central Intelligence System
Device Stabilizer
Scratch Resistant Outer Shell
Left Wheel

Figure 3.4.7 Block Diagram of Potential Design

In this design the entirety of our electronic hardware and sensor devices would be encapsulated within a hard protective cylinder exoskeleton. This structural design would ensure that the hardware was isolated from external forces. We decided not to use this and instead go with the Spherical exoskeleton of Figure 3.4.2 and Figure 3.4.3 However, in prototyping this model we found the Central Intelligence System frame right wheel and left wheel could be incorporated into our new Spherical Design. In essence, by putting all of the hardware within a protective Sphere we would defend from the cat's predatory attacks invoked by our kitty-bot movement algorithms. Secondly, the Power Supply, Device Stabilizer, Sensors and Central Intelligence system would all be placed within close physical proximities of each other in a stable position. This meant that when prototyping our Spherical design, we could use the useful parts from the old design so that we would not have to work twice. This would much simplify our electrical design in the future we thought, which was later confirmed in our rapid prototyping stage. We would also create a frame to hold all of these components that would be used in our Sphere. This frame would have to have two rods attached where the wheels used to be to direction the Sphere and propel it. In our original prototype each wheel would be controlled by a motor and would make decisions based our system of microcontrollers which would later be interwoven into a PCB design which we called our Central Intelligence System. The device stabilizer and sensors would as well be programed to react to input data or take data in and analyze it based on this system. In our new system all we had to do is replace the wheel with a rod that attaches to the inner part of either hemisphere and we could keep all of our work from the first prototype. This of course made us happy because we saved time. One of the drawbacks our original rapid prototype design was that we would have to program sensors to recognize when our devices has be knocked over and needs to be re-stabilized. These sensors would also have to recognize when the device is upright so as to not tip it back over by the Device Stabilizer. The device stabilizer itself which would be constructed as somewhat of a L shaped rigid tail that spins when the robot senses that the unit has been destabilized. The spinning motion of the device stabilizer would make it so that the L shape would make contact with the ground and the force between itself and the ground would propel our unit to an upright position. The point is, with our new Design we did not have to consider any of this. Our spherical structure made it so that our device would never be knocked down.

3.4.7 Why we chose the Spherical Design over the Cylindrical Design.

Our cylinder based design have other consequences we were uncomfortable with. In order to further consider them we modeled our ideas. By 3d modeling our ideas we were able to critically think through potential design flaws and or advantages. This model was created and referenced so that we could see what the structural architecture would cause us to consider in terms hardware design. In this rough AutoCAD 3D model sketch we noticed several things that we would have to overcome. Firstly, notice the green appendage in **Figure 3.4.9.** We called this item the device stabilizer. In order for this structural architecture to work the Device Stabilizer would have to be promptly adjusted to the rest of the unit. The device stabilizer would rotate clockwise or counterclockwise in the event our Kitty-Bot was knocked over by the playful feline. This means we would have to add another motor to the unit which would further increase overall complexity of the Kitty-Bot. Also the motor would have to be strong enough to propel the entire system upright in the event it was knocked over. This in turn, would mean we would have a further power consideration variable to add to our ever increasing complexity equation. With the Spherical design we would not have to consider any of these difficulties.

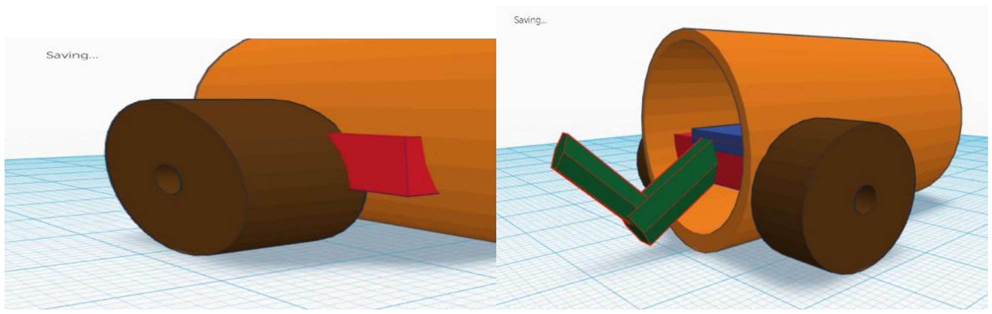


Figure 3.4.8 3D Model of Cylindrical Design

Secondly, when trying to 3D model our our Structural Design we found ourselves having a hard time deciding how to design the wheels and the motors that attached to these wheels. Functionally, the way we mentally imagined things working before the 3D model was our central intelligence unit controlling the servo motors which would in turn spin the wheel. However, our Cylindrical structural design made it so that that would be difficult.

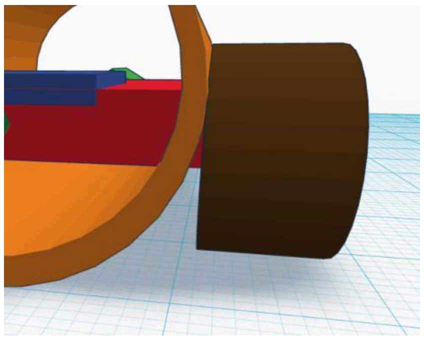


Figure 3.4.9 View of Cylindrical Design's Wheel

The 3D model made us notice that we would have to cut into the Cylinder in such a way that the wheels would not touch it while simultaneously keeping the hardware encapsulated. Also, after further considering this design we became concerned about the safety of the animal. The point of this device is to have a cat attack it from as many angles as possible. This being said we imagined a scenario where the cat would try to attack the wheel of our robot, get a nail stuck in one of the moving parts, specifically one of the motors or wheels, and potentially get hurt. This, of course was a grave concern for our team and highly weighed on the negative end of the scale in terms of our final judgment of this structural design.

Next we considered what our encapsulated parts would look like in this design. One of the main selling point to further consider, 3D model and evaluate this structural design was the Cylinder outer shell that protected all of the hardware on the inside. **Figure 3.4.10** shows the very basic 3D model we used to further explore this design. However, with the Spherical design we had an even better and more complete encapsulation of all our hardware.

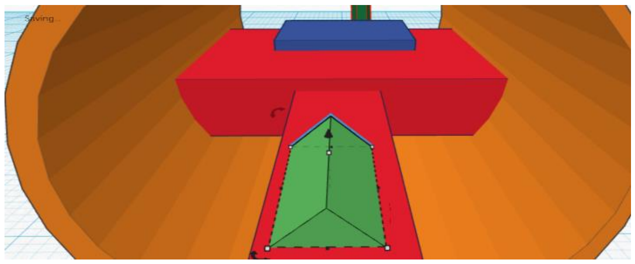


Figure 3.4.10 Internal View of Cylindrical Design

The Blue and red boxes in this figure represent the Central Intelligence unit, any electrical connections to it, the physical structure holding this including the connections to the two motors controlling the wheels outside the cylinder and the battery pack. The green elongated triangular pyramid represents the location where we would have decided to place our sensors. We would have decide to place our sensors here because of several reasons. Firstly, concentrating all the sensors at the front of any unit seems to be the best solution evolution came up with so we figured we would copy nature. In essence we wanted to mimic some kind of counterfeit rodent or prey that the cat would pounce on. The prey would react to the cat based on algorithms we programed into it. The sensors in the front of our Kitty-Bot would act as the input to our Central Intelligence System that would then make decisions that would further incite the cat's playful nature based on our coding algorithms. The sensor system was one of the few advantages the Cylindrical system had over the Spherical system. However, the advantages never outweighed the disadvantages.

Our 3D modeling also made it clear that that the Cylinders encapsulation of our hardware would perhaps not be as thorough as one would like. As we all know cats can be very creative in their mischief. It would be very easy for the cat to stick its paw inside the Cylinder and cause a destructive force to be applied towards our hardware, power supply and electronics. This would, of course, also cause a great safety concern to the animal. If the cat stuck its paw inside the cylinder and got its nail stuck or even worse somehow managed to electrocute itself this would be an extremely unfortunate event. In the end this candidate had many virtues, but also many faults.

It was made clear to us after modeling the structural design that this candidate would have to overcome the shortcomings an undesirable amount of the design flaws. Further design would be necessary to overcome the safety issues embedded in this structure. Perhaps, we reasoned, we could have some kind of netting covering the back and front ends of the cylinder. However, if we did choose that approach we would then have to find a way to let the sensors still peek through. Maybe a mesh of some sort. Overcoming the safety issues revolving the motors and wheels however was more challenging to think about. At best we could make the wheels have hub caps that covered its dangerous parts. However, if the unit was tilted over these dangerous parts would still be exposed. Other concerns the team had related to the Device Stabilizer needing substantial torque. The force created by this torque would means the Device Stabilizer would be prone to break. Also, the Device Stabilizer itself could potentially be a risky element for the predator cat. This cylindrical dual motor controlled wheel navigation design was the first thing we came up with. However, after imagining the design and its challenges in the more mature versions of its prototype we became even more confident in our decision to choose the Spherical Design.

3.4.8 Creating a Second Robot

Our decision to implement a spherical design solved many issues, but it also presented new challenges. We soon discovered that with a spherical robot it would be very challenging to implement sensors. All the sensors we considered presented challenges when it came to attaching them to a spherical robot. No sensor could be attached to the outside shell as that shell would be constantly rotating, meaning no good readings could be taken, not to mention possible damage to the sensor. What about placing them inside? With an internal counterweight some level of stability could be applied to central chassis. Whether it would be stable enough is questionable, but there loomed a bigger problem with this idea. No sensor would be able to work properly through the inside of the shell. Ultrasonic sensors would constantly go off because sound wave would just bounce off the inside of the shell and trigger. Any type of light sensor would not pass through a solid shell, and if the shell was transparent the light signal would be distorted. Another type of sensor we considered was a touch sensor. Since they just register contact, we wouldn't need to worry about line-of-sight with this sensor. They could potentially be affixed to the shell and register any external physical contact with the shell. Whether or not the rotation of the sphere itself would set off the sensor would need testing. Besides that potential roadblock is a much bigger one, the wiring. The sensors would have to be wired to the control unit in the central chassis, and the best place to put the touch sensors would be on the sphere. If the sensors are on the sphere, which is constantly rotating, the wiring would get tangled around the center chassis.

Implementing sensors that allow the robot to respond to an external stimulus was a part of our requirements so we had to create a solution. Our solution became to create a second robot. Utilizing elements from our original design, this second robot would take on a more traditional design, akin to that of a bo-bot, with two motorized wheels and third unpowered wheel for stability. With this type of design, we now had the freedom to attach sensors. We decided upon an ultrasonic sensor. When the sensor detects an object, a “meow” sound plays through an attached 8-ohm speaker.

Playing audio presented us with unique challenges. Usually playing audio files requires them to be stored. Since most microcontrollers have low memory, especially the TI MSP430, external such as an SD card is needed. Considering the limitations of the MSP430, we decided to implement an Arduino Uno for this task instead. Our original circuit design for the Arduino is shown in **Figure 3.4.11**.

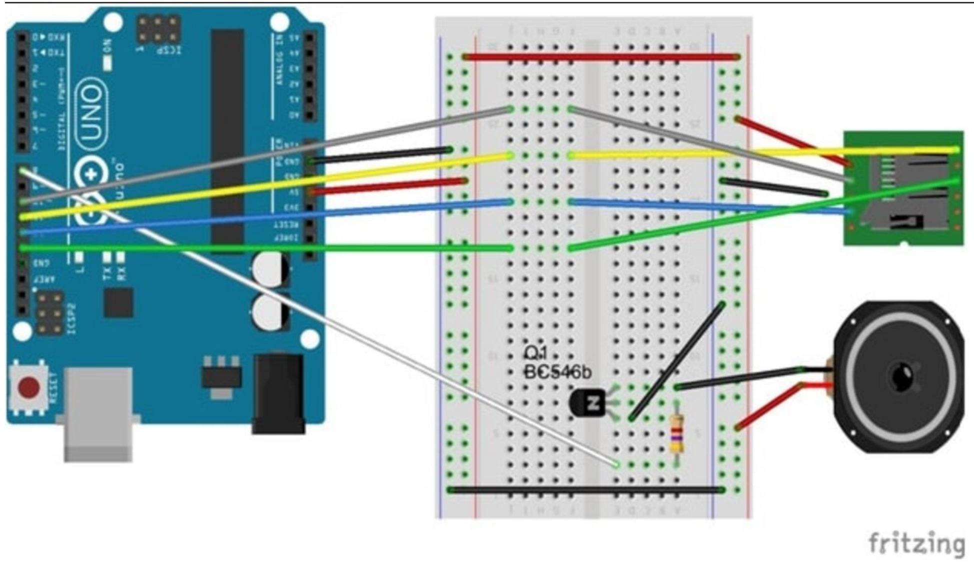
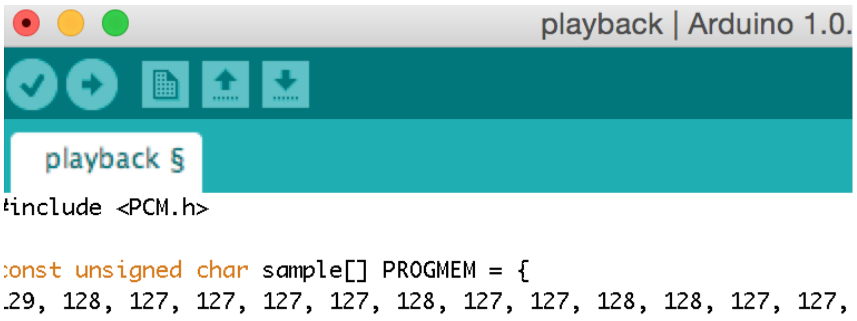


Figure 3.4.11 Original Arduino Circuit

As you can see in this circuit, we still thought we needed an SD card to store the audio file. This would require more components which brings more complications. Instead we found another solution that would simplify our circuit. Since our sound effect was meant to be a short meow sound it wouldn’t be that large of a file, so we down-sampled the audio to a short playback of about 4 second with a low-bitrate (8 KHz) to increase space. Inside the custom settings, select “Stereo Bit Rate” of 16 KHz, a “Sample Rate” of 8 KHz, and Mono “Channels”. Then, it was converted to a series of numbers that could be passed into the Arduino program **Figure 3.4.12**.



```
‡include <PCM.h>

:onst unsigned char sample[] PROGMEM = {
.29, 128, 127, 127, 127, 127, 128, 127, 127, 128, 128, 127, 127,
```

Figure 3.4.12 Number Encoding for Audio

This robot needed the use of two microcontrollers. Due to the libraries we wanted to use for our parts, we needed two separate clocks, one for the motors and one for the ultrasonic sensor. Two microcontrollers were the quickest solution due to time constraints. Our final circuit for this robot is shown in **Figure 3.4.13**.

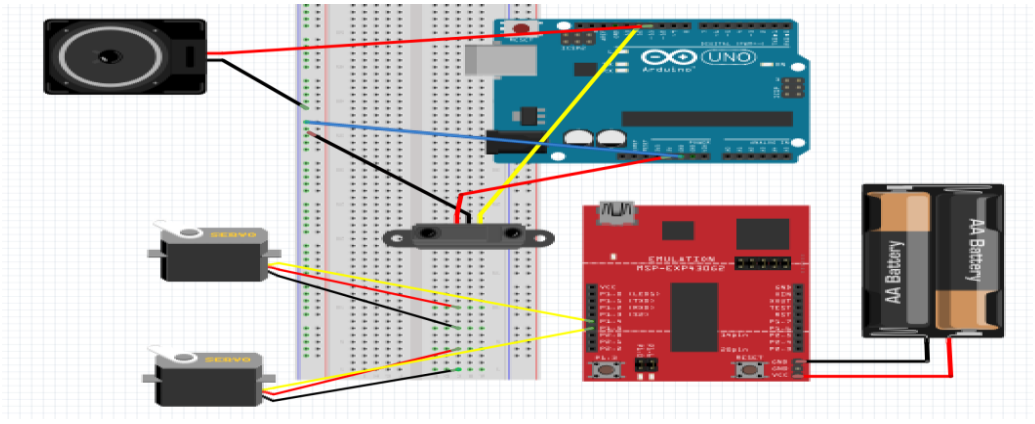


Figure 3.4.13 2nd Robot Circuit

Order to fit all of our components we wanted to try a 3D printed chassis. We modeled a chassis in SolidWorks, **Figure 3.4.14**.

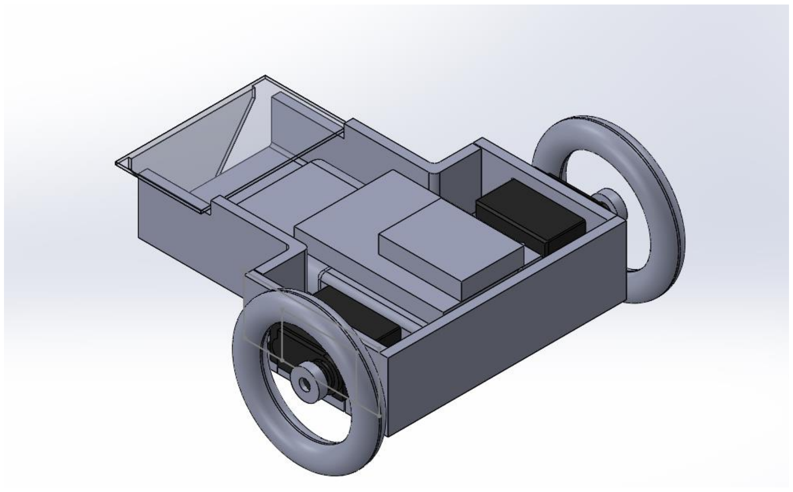


Figure 3.4.14 SolidWorks 3D Model

This model was then printed, taking a total of 10 hours to complete. Once finished printing, we assembled the chassis which was in 4 parts, the front wall, the two side wheel walls, and the overall base. Once assembled, the components were placed inside. The 3D model was designed in a way to fit everything tightly without wasting any space. For this prototype, the wheel we utilized was the third, ball-like wheel of a bo-bot. The final prototype of this robot is shown in **Figure 3.4.15.**

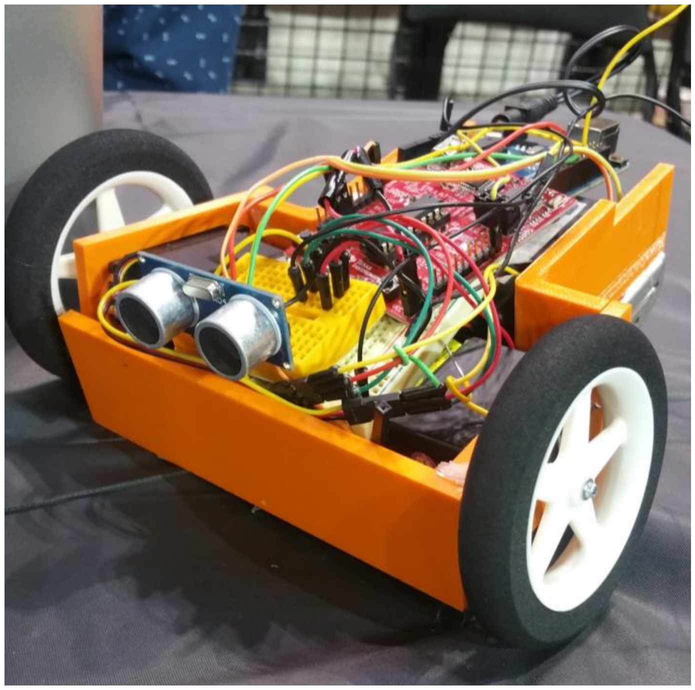


Figure 3.4.15 Second Robot Prototype

4 Identification and Review of Applicable Standards

4.1 Research and Identification of Standards

4.1.1 Research on Standards

Engineering standards are documents that specify characteristics and technical details that must be met by the products, systems and processes that are being developed. These include details such as dimensions, safety aspects and performance requirements. The purpose of developing and adhering to standards is to ensure minimum performance, meet safety requirements, make sure that the product/system/process is consistent and repeatable, and can ensure compatibility with other standard-compliant equipment. A code is a law or regulation that specifies minimum standards to protect public safety and health such as codes for construction of buildings.

Standards may be referenced or included in the specifications, which are a set of conditions and requirements of precise and limited application that provide a detailed description of a procedure, product or service for use primarily in procurement and manufacturing.

Our project must follow certain standards, codes and requirements in order to be able to be developed and deemed safe. As it stands, we must meet certain requirements in order to ensure the security of ourselves, those around us while demo-ing and our test subject, the kitten. We must make sure sensitive parts aren't exposed that the kitten could get into and end up electrocuting itself. Our PCB and protective circuits must remain under a certain temperature as to not overheat causing product failure and/or a fire.

4.1.2 Identification of Applicable Standards

IEC 61249-2-23 Ed. 1.0 b:2005

Title: Materials for printed boards and other interconnecting structures - Part 2-23: Reinforced base materials, clad and unclad - Non-halogenated phenolic cellulose paper reinforced laminated sheets, economic grade, copper clad"
Scope: This part of IEC 61249 gives requirements for properties of non-halogenated phenolic cellulose paper copper-clad laminated sheets, economic grade, in thicknesses of 0,8 mm up to 3,2 mm. This standard covers material with different requirements on flammability and is designated according to the following: Material 61249-2-23-1: general purpose grade, requirement on flammability not specified; Material 61249-2-23-2: materials of defined flammability (vertical burning test). These grades of material provide for one of two flammability requirements and designated as FV0 or FV1.

IEC/TS 62657-1 Ed. 1.0 en:2014

Title: Industrial communication networks - Wireless communication networks - Part 1: Wireless communication requirements and spectrum considerations
IEC TS 62657-1:2014 (en) provides the wireless communication requirements dictated by the applications of wireless communication systems in industrial automation, and requirements of related context. The requirements are specified

in a way that is independent of the wireless technology employed. The requirements are described in detail and in such a way as to be understood by a large audience, including readers who are not familiar with the industry applications. Social aspects, environmental aspects, health aspects and market requirements for wireless communication systems in industrial automation are described to justify the wireless communication requirements. This Technical Specification describes requirements of the industrial automation applications that can be used to ask for additional dedicated, worldwide unique spectrum. This additional spectrum is intended to be used for additional wireless applications while continuing using the current ISM bands.

CISPR/TR 28 Ed. 1.0 b:1997

Title: Industrial, scientific and medical equipment (ISM) - Guidelines for emission levels within the bands designated by the ITU (International Telecommunication Union)
This technical report provides the guidelines for emission levels within the bands designated by the International Telecommunication Union (ITU) for industrial, scientific and medical (ISM) application.

IEC 62115 ED. 1.0 b:2011

Title: Electric toys – Safety – Deals with the safety of toys that have at least one function dependent on electricity. Examples of toys within the scope of this standard are constructional sets; experimental sets; functional toys (having a function similar to an appliance or installation used by adults) and video toys (toys having a screen and means of activation, such as a joystick or keyboard.

4.2 Design Impact of Relevant Standards

A quick review of the standards above shows that this project is not directly affected by many standards, however, it is subject to a few. The first standard that is relevant is IEC 61249-2-23 Ed. 1.0 b:2005 which is a standard that governs flammability characteristics of PCB materials. This is obviously important to the project since we employ a multi-layer PCB in our project. The board must conform to this standard since it is built using industrial materials. The next two documents are used to set standards for wireless communication. Our project has wireless communication using the HC-05 Bluetooth module, and it turns out to be a good fit because it conforms to the FCC standards listed as well as the ITU standard.

The final standard listed is related to electric toys. Technically, our project is a toy (for kittens) and it runs on electricity, therefore, this standard is applicable to our design. Even though the robot is being made for kittens to play with, it is being operated and maintained by humans, so it must meet this standard which is necessary for it to be developed. Our protective circuits must be secure so that wires or other components will not be exposed which can cause electrical burns and/or electrocution to the human handling Kittybot or the kitten that is playing with it.

5 Realistic Design Constraints

The following sections will discuss possible challenges faced in the development of this project. These include monetary, time, and environmental constraints, as well as the factors relating to health, safety, and practical use.

5.1 Economic and Time Constraints

The project's total cost will be relatively low. Based off of our estimations the project shouldn't exceed $500 in total cost. It should fall around $250 to $300. The project must be completed by December of 2016. We will begin rapid prototyping with old parts and personal items we previously owned as early as June 2016 to help during the designing phase. This will help in the building phase as well. After the rapid prototyping and design is near completion we can see what components can be salvaged from the prototype for use in the final build of KittyBot. With previously owned components we are saving time and money. However, some parts will eventually need to be ordered. While this obviously presents an economic constraint; that is planned for, we priced items and parts in our initial report. A more nebulous problem is possible delivery time for certain parts. Items that can be found in local retail and hardware stores (Walmart, Home Depot, Pet Supermarket, etc.) can be occurred anytime if the item is in stock. We purchased a hamster ball from a local Petland pet store to test early prototypes with; and hardware items such as screws, nuts, and polyvinyl chloride (PVC) piping can be purchased at store like Home Depot and Lowe's. Some other items may only be available online and with this comes potential complications when it comes to shipping times. The best case scenario for online ordering would be if the desired item was available on Amazon. More specifically, since we have access to an Amazon Prime account, if the item is available under the Amazon Prime banner we can get free two-day shipping. A small calibration was purchased for early prototype testing using this service. Amazon Prime has a continuously expanding library of goods, but some of the more sensitive electronics needed for KittyBot will not be available with the Prime service. Most other goods providers will not be able to provide free shipping, and if it is free it will most likely be very slow delivery ranging from one to three weeks. Having to wait long periods of time for components can halt development, cause milestones to be missed, and deadlines to be pushed back. Getting faster delivery would then mean paying more for express shipping, thereby increasing the costs. The time constraint of shipping can become an economic constraint as well.

5.2 Environmental, Social, and Political Constraints

KittyBot will be meant for interaction with indoor cats. It will move best on smooth surfaces like wood or tile floors and even low carpet, basically common flooring surfaces found in most everyday homes. KittyBot is not meant to be used outdoors; i. It is not recommended to operate the device in grass, sand, mud, snow, etc. because it is not designed as an all-terrain device. Also it will not be water-proof, water-tight, or water-resistant and will malfunction if allowed to is soaked or submerged in water. Cats are meant to interact with KittyBot, so it will be able to withstand potential damages from household cats. Animals comparable to the size of a cat could also interact with KittyBot, certain breeds of small dogs for instance. Larger animals like large dogs could cause damage to KittyBot; as such, it is not recommended for these animals to play with the device.

5.3 Ethical, Health, and Safety Constraints

Since KittyBot is meant to interact with pets we have a responsibility to design and construct KittyBot in a manner that will in no way harm the animal. The casing housing all internal electronics for KittyBot will be smooth with no sharp edges that could cause scrapes or lacerations to the intended animals. KittyBot will be of a weight and size that shouldn’t hurt or crush the animal if bumped into. Again, cats are the intended audience for play with this device. If smaller animals like hamsters, mice, lizards, or snakes are allowed to interact with the device they could potentially be harmed by its size and weight. On the other hand, animals larger than cats could also potentially be harmed KittyBot. If a large dog were too rough and cracked open the device, they could cut themselves on the fractured shell or damaged electronics. This is a pet toy, as such it not meant for use with children. Depending on the age of the child they could break KittyBot and cut themselves or even swallow any broken off pieces of the device. The best course of action for any user of our device is to always maintain adult supervision when allowing your pets or children to play with KittyBot.

5.4 Manufacturability and Sustainability Constraints

Kitty-Bot will be made from majority inexpensive components, with the most expensive single piece being the printed circuit board. The structural components are all inexpensive. The outer shell is a plastic hollow ball. A common hamster ball found in most pet stores for under $10 would work just fine for this casing. An internal chassis will be 3D printed. Fortunately for our group, as University of Central Florida students we have access to free 3D printing. Since this will be made from acrylonitrile butadiene styrene, or ABS, plastic which is one of the cheapest 3D printing materials, and it will be no larger than 10 cm in width, length, or height, this chassis can be cheaply reproduced. Any additional screws, bolts, or brackets for construction are all inexpensive items. Microcontrollers and sensors are price comparable to the printed circuit board while still not being as expensive. These electronics fall within the $20 to $40 range. Motors can also be around $30 a piece. Batteries plus potential cases and battery holders can total upwards of $20. With the right combination of components, found at the right prices, KittyBot could be manufactured for well under $150.

When it comes to long term sustainability, that can be difficult to determine exactly. Starting from the outside, the integrity of the outer shell should hold for quite some time, years in fact, if operated in the intended ways. KittyBot should be used to play with cats primarily, so it can withstand regular play from cats. With the average number of cats in a single household being two or three, KittyBot shouldn’t be overwhelmed by the number of feline participants in the vast majority of households. KittyBot may ram into walls or other objects. For the top speeds KittyBot will be able to reach, this ramming should cause no significant damage after extended periods of time. When it comes to the other structural components, the screws, nuts, and brackets holding KittyBot together should all be able to withstand a cat's activity as well as bumping into things. The overall structural integrity is not meant to be able to take activity beyond this. As stated early, larger pets such as large dogs are not recommended for KittyBot. These types of pets may be able to damage KittyBot with their paws, bodyweight, or jaws. Unsupervised children of a certain age can pose a danger to KittyBot's structure as well. KittyBot can withstand bumping into objects while moving at its top speed but acts that can cause KittyBot to collide into objects at speeds surpassing that like rolling, throwing, or dropping KittyBot can most certainly cause severe damage to the structural elements of the project.

The internal elements of the KittyBot are much more sensitive. Putting these components inside an outer casing can will help in keeping them safe. The mechanisms and components fastening the electronics in place will hold under the intended uses of KittyBot. Unintended use can cause damage to these internal components as well. Excessive force can knock the central chassis out of place, or at worst the shocks can cause damage to the printed circuit board or microcontroller. Extreme heat and sun exposure, extreme cold or snow, and rain and water can cause danger. An indoor environment is best. KittyBot should be operated in temperatures ranging from 65 to 75 degrees Fahrenheit.

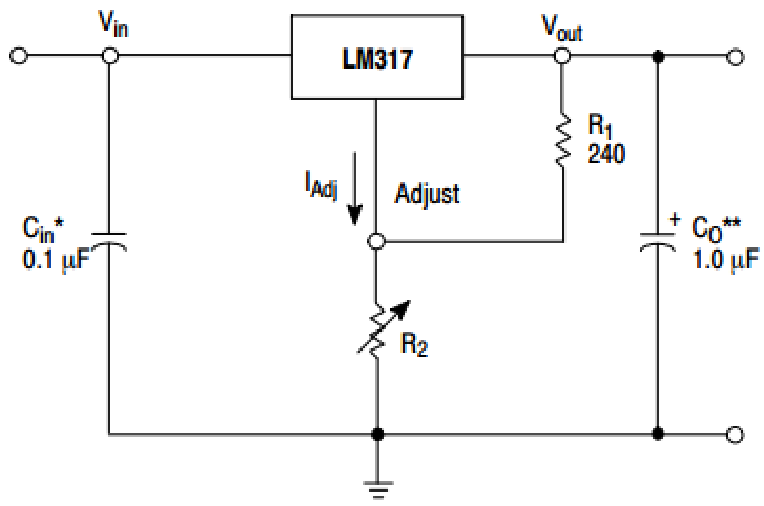
6 Hardware Design

The kitty-bot is made using both mechanical and electrical hardware. These systems need to work together through sensors, motors, embedded systems, printed circuit boards, and microprocessors. The power supply systems need to accommodate all these hardware components with enough power to run the kitty-bot correctly.

6.1 Voltage Regulation

Voltage regulators are important in electrical systems. They allow systems to run 12V batteries without out sending too much power to more sensitive components. Voltage regulators don't use much power due to the tiny current that runs through it. The servo motors could require a larger voltage to power them and therefore a voltage regulator will be needed to regulate voltage for the microcontroller and sensors. Having a large gap between V_{in} and V_{out} causes inefficiency in the regulator.

Capacitors are used to filter noise input prior to regulation. Another capacitor is placed on the output of the voltage regulator to deal with voltage spikes from the regulator on the load resistor. Different sized capacitors are used in the figure below to filter different ranges of frequency. The smaller capacitor prevents large frequency signals to interfere with the motor. **Figure 6.1.1**



* C_{in} is required if regulator is located an appreciable distance from power supply filter.
** C_O is not needed for stability, however, it does improve transient response.

Figure 6.1.1

6.1.1 Linear Regulators

A linear voltage regulator is a technological device that acquires a certain amount of input voltage and then helps to govern a predetermined amount output voltage. They are advantageous when being applied to a circuit board due to their compact size and ability to produce both positive and negative output voltages. Linear voltage regulators come in multiple voltage types, such as either fixed voltages or adjustable voltages. An example of a fixed voltage linear regulator is the 78xx series. An example of an adjustable voltage linear regulator is shown in LM317, below in **Figure 6.1.2.**

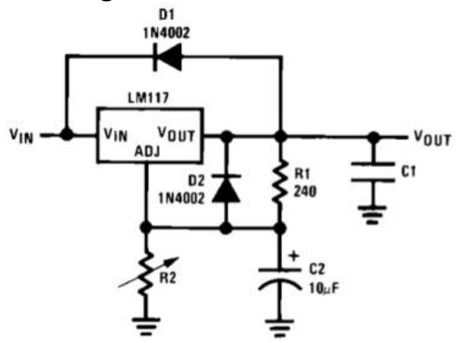


Figure 6.1.2

(From www.ti.com)

Our team has found linear regulators that can provide output voltages ranging from 1 to 40 volts. The current from the load is at less than 1 to 1.5 amperes. However, there are input voltage requirements for linear voltage regulators. The input voltage must fall between a minimum and a maximum range. The minimum voltage requirement is determined by the dropout voltage, which is found on the datasheet of the linear regulator. This dropout voltage is generally between 2-3 volts. Therefore, a 10 volt regulator would usually have a minimum voltage requirement of 12-13 volts. The maximum input voltage is dependent on the part selected and generally yields up to 40 volts.

For KittyBot, a linear voltage regulator is an adequate piece of technology. Our required voltage level falls within the range of the minimum and maximum requirements of a linear voltage regulator. The ampere level is thoroughly covered by the average level of a linear voltage regulator and thus is more than within reason. Our team's battery will provide input voltages at a considerably higher level than the desired regulator output. Therefore, the dropout voltage requirement will also be met. Linear regulators meet the requirements of our team's project and in conclusion can be used for most voltage regulation needs.

Another advantage of a linear voltage regulator is the practicality in its design. The voltage regulation is performed entirely within KittyBot's integrated circuit. Therefore, no add-ons are needed when implementing the regulator. The linear regulator usually has 3 pins, which allows it to be applied easily to KittyBot's printed circuit board. However, there is one disadvantage of a linear voltage regulator: its efficiency. There can be certain cases where the input voltage is much higher than the regulated output voltage, in which the linear regulator is inefficient in trying to banish the extra power as heat. For this project, the input voltage will be 5V typically which is not a large drop off to supply the 3.3 V needed for the microcontroller therefore it will be approximately 90% efficient.

6.1.2 Switching Regulator

Switching regulators have the ability to take a higher voltage and it brings it down to a lower voltage as well as linear regulators. Switching regulators have the ability to transform a lower voltage to a higher voltage. Switching regulators can also be very efficient with their power output, looking at some datasheets we saw that some regulators have an efficiency of 90-95%. Switching regulators also create very little heat when being used.

The downsides of using the switching regulators are that they cost more and are little more complex. Another downside is that switching regulators are noisier than linear regulators. Another obstacle switching regulators present is that they tend to have an output current greater than 1 amp. The regulator will be powering our microcontroller which has a very low input current. The MSP430 has an input current in the micro amp range.

Switching regulators have three different abilities which have names they are common for. Those three names they're known for are buck (step down), boost (step up), and buck-boost (step up/step down) regulators. Our battery will be supplying anywhere from 5-6 volts to loads which require a range of 3.3 to 5 Volts. A buck regulator was found courtesy of Texas Instruments, they had a voltage regulator which met our group's specifications. The LM2594 step-down buck converter is an adjustable regulator which works well with battery applications. The pin level diagram of this chip is pictured below in **Figure 6.1.3.**

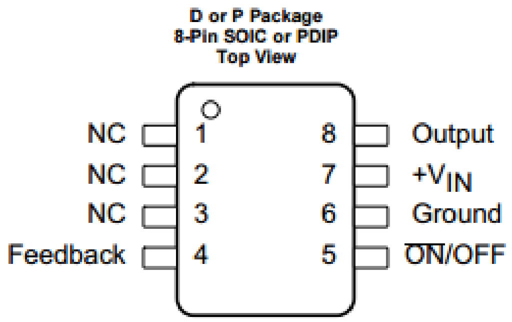
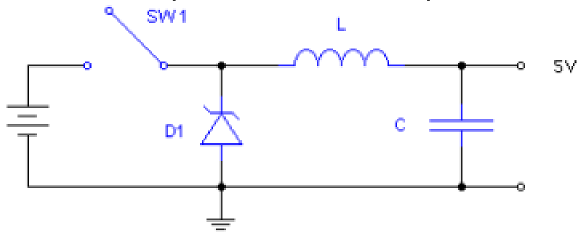


Figure 6.1.3

(From www.ti.com)



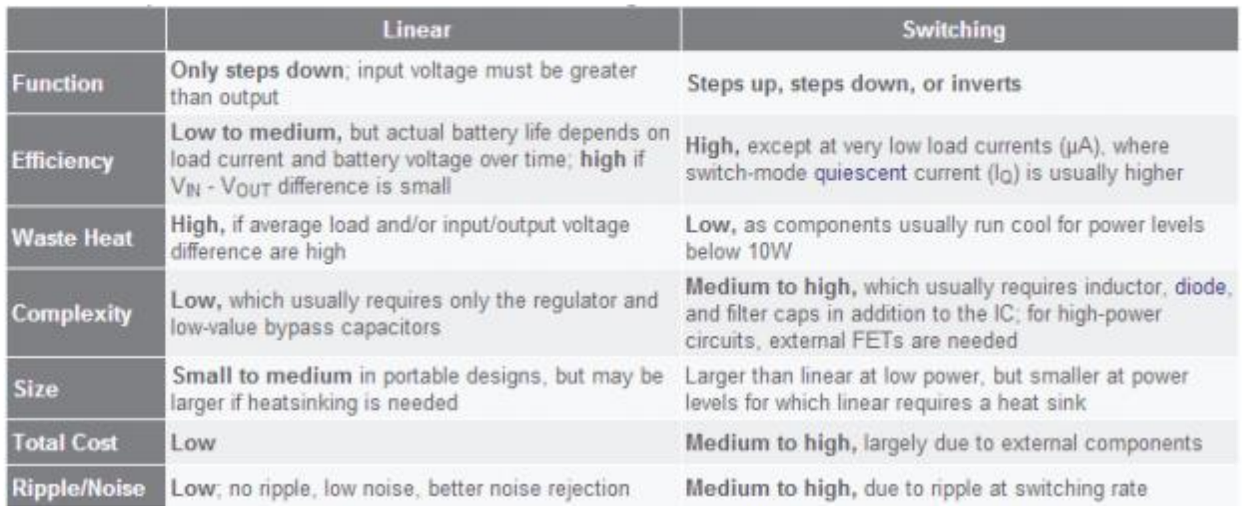
Switching voltage regulators use a combination of transistors acting as switches and inductors or capacitors acting as storage devices to provide a constant output voltage. Switching regulators can further be divided into categories such as buck, boost, and buck/boost regulators. A buck regulator takes a higher input voltage and steps it down to a constant lower output voltage. For this project, a buck type regulator will be required. Switching regulators are available as complete integrated circuits just like linear regulators. Typically used parts handle supply voltages of up to 40 volts or higher and can handle currents up to about 3 amperes. Switching regulators do not convert the difference in power to heat like linear regulators and therefore have power efficiencies of up to 95%.

6.1.3 Regulator Trade-offs

The benefits of using switching regulators over linear regulators are mainly because of their power efficiency. KittyBot needs to be efficient in order to work for as long as possible. Switching regulators are also available online as complete integrated circuits and would therefore be easy to integrate and implement into KittyBot's circuit design. Switching regulators have the ability to step down or up giving more versatility, but for the KittyBot only step down is needed. Switching regulators produce very little heat compared to a linear regulator.

There are a few key advantages linear regulators have over switching regulators however. Linear regulators are less noisy because switching regulators can produce electric interference due to their utilization of inductors and can have a ripple voltage. Ripple voltage can also be caused from a high switching rate. The switching in the regulator causes it to be much noisier than its linear counterpart. Linear regulators benefit from being smaller due to switching regulators requiring additional components to build the desired switching regulator circuit. Although linear regulators can be quite large if a heatsink is necessary. This design's case wouldn't require a heatsink.

This project would be able to meet the requirements set out for KittyBot with either a switching regulator or a linear regulator. The input power supply and output power requirements fit into the specifications of readily available switching regulator parts. Many of today's electronic devices using microprocessors also use switching regulators. This makes finding existing circuit designs easy and allows us to be able to change the designs to help benefit this project. Linear regulators can only regulate voltage lower but for the design, as mentioned before, they still work out. **Figure 6.1.4** was created below to see the advantages or disadvantages between the regulators easier



	Linear	Switching
Function	**Only steps down**; input voltage must be greater than output	**Steps up, steps down, or inverts**
Efficiency	**Low to medium,** but actual battery life depends on load current and battery voltage over time; **high** if V_{IN} - V_{OUT} difference is small	**High,** except at very low load currents (μA), where switch-mode quiescent current (I_Q) is usually higher
Waste Heat	**High,** if average load and/or input/output voltage difference are high	**Low,** as components usually run cool for power levels below 10W
Complexity	**Low,** which usually requires only the regulator and low-value bypass capacitors	**Medium to high,** which usually requires inductor, diode, and filter caps in addition to the IC; for high-power circuits, external FETs are needed
Size	**Small to medium** in portable designs, but may be larger if heatsinking is needed	Larger than linear at low power, but smaller at power levels for which linear requires a heat sink
Total Cost	Low	**Medium to high,** largely due to external components
Ripple/Noise	**Low**; no ripple, low noise, better noise rejection	**Medium to high,** due to ripple at switching rate

Figure 6.1.4

(Used from Digikey.com)

6.2 Amplifier Circuit Design

Amplifier

One of the reasons we wanted to design a decent amplifier is because. For example, our censor output voltage range might not match up well with the msp430. Also the msp430 might not output enough current to correctly power the motors. Being that we want to avoid these issues we wanted to design an amplifier that would give us flexibility. We wanted to put the input values into a computer program that would give us the output values desired, and that's exactly what we did. There are advantages and disadvantages, however. For example, a common emitter amplifier might have low impedance and is inverting. These are all things that we would like. These are characteristics of, for example, a common emitter amplifier. The high output impedance and current gain might be ok as well. However, the high voltage gain is what we really desire. That will improve the quality of our device. We figured an average voltage for the circuit board that we want to provide power supply to might be two volts. We would want an input resistance of about 15,000 ohms and an output resistance of about 100 ohms.

At the end we decided a multi stage Common emitter amplifier would be our best bet. Our major point of focus was a high gain. After a lot of math and a very crafty computer program written in C we arrived at the following specs.

Design Procedure:

1. To design the amplifier circuit, we need to break the specifications down into two parts: a high input resistance that produces a large gain, and a low output resistance. Two amplifier circuits fit the bill only somewhat. A common emitter amplifier can amplify the voltage by a great factor, but moves the phase of the output such that it is 180 degrees out of sync with the input - basically, inverting the output along the real axis. A common collector has a low output resistance, but does not negate the phase shift of the output. As such, a third component is added: a common emitter amplifier with no bypass capacitor.

2. To calculate the proper values, the circuit must first be transcribed into its DC and small signal equivalent circuits. For practicality, capacitors are taken as relatively large values between 10 micro farads and 100 micro farads. When the impedances of these components are calculated, these values will result in a short circuit when considering small signal values, and an open circuit for direct current values. Therefore, the equivalent circuits can be shown in **Figure 6.2.1** and **Figure 6.2.2**.

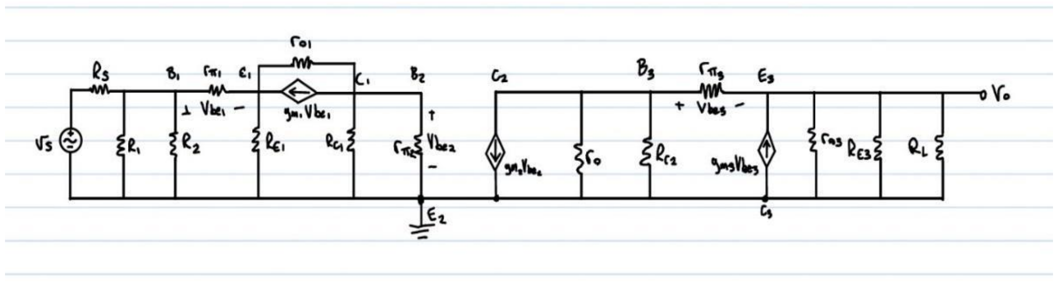


Figure 6.2.1 Circuit diagram of Small Signal equivalent of the amplifier circuit

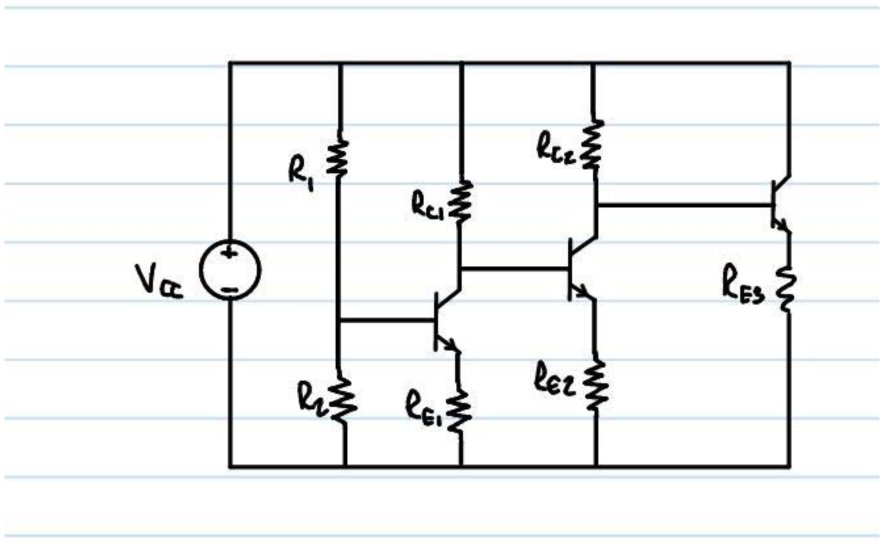
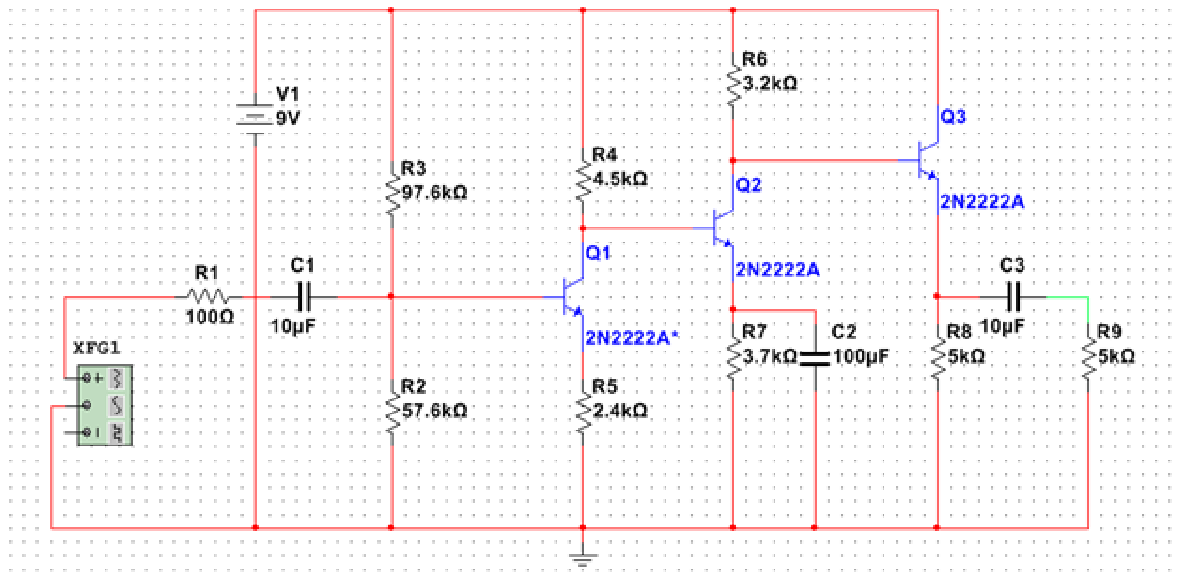


Figure 6.2.2 Diagram showing the DC equivalent circuit of the amplifier

3. From the above circuits, one can estimate several resistance values based on the requirements set forth in the lab. Once these values are obtained, the rest of the resistance values can be obtained through calculation and estimation.

4. In order to simplify the construction of the circuit, the obtained values can be rounded to the nearest available physical resistor value.



After analyzing the circuit, these equations which were used to bias the circuit.

Stage 2 and Stage 3

RE3 = (Vcc - VCE3) / IC3;

RC2 = (Vcc - VBE - IC3*RE3) / (IC2 + IC3/B);

RE2 = (Vcc - RC2*(IC2 + IC3/B) - VCE2) / IC2;

RC1 = (Vcc - IC2*RE2 - VBE) / (IC1 + IC2/B);

RE1 = (Vcc - RC1*(IC1 + IC2/B) - VCE1) / IC1;

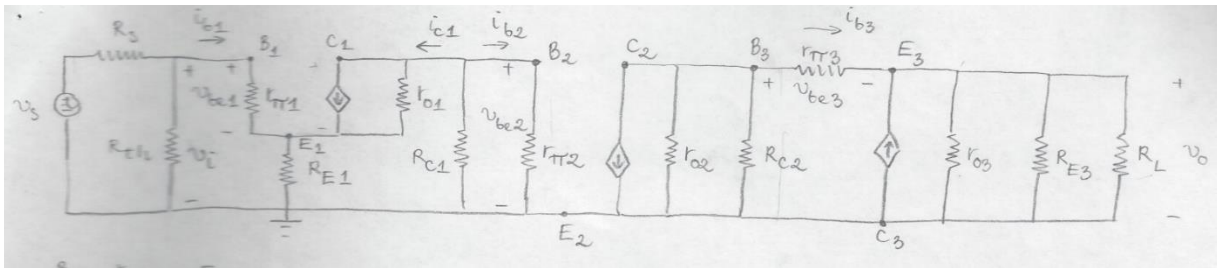
Stage 1

Rth = 0.1*(1+B)*RE1
Vth = (Rth/B + RE1)*IC1 + VBE

R1 = Vcc*Rth/Vth;
R2 = Vth*R1/(Vcc - Vth);

Rib = rpi1 + (1+B)*RE1;
Ri = 1/(1/Rib + 1/Rth);

Small Signal



Derived Small Signal Equations

• rpi1 = B*VT/IC1; • rpi2 = B*VT/IC2; • rpi3 = B*VT/IC3; • ro1 = VA/IC1; • ro2 = VA/IC2; • ro3 = VA/IC3;	• gm1 = IC1/VT; • gm2 = IC2/VT; • gm3 = IC3/VT; • Rib = rpi1 + (1+B)*RE1; • Ri = 1/(1/Rib + 1/Rth);

After trying several values these worked best**.**

Input values

IC1 = 1mA
IC2 = 1mA
IC3 = 1mA
VCE1 = 2V
VCE2 = 2V
VCE3 = 4V
RL = 5K
RS = 100

Resulting Values A

• RC1 = 4569.536621 • RC2 = 3278.145508 • RE1 = 2399.999512 • RE2 = 3699.999512 • RE3 = 5000.000000	• Rth = 36239.992188 • Vth = 3.341599 • R1 = 97605.937500 • R2 = 57641.718750 • Rib = 366299.937500 • vbe2 = -0.864712 • ib3 = 0.000281

Resulting Values B

• VCE1 = 2.000000 • VCE2 = 2.000000 • VCE3 = 4.000000 • RAC1*IC1 = 4.504151 • RAC2*IC2 = 3.249525 • RAC3*IC3 = 2.455285	• Ri = 32977.367188 • Ro = 46.391895 • Gain = 103.575066 • Maximum unclipped voltage = 2.000000

6.3 Embedded System

6.3.1 Microcontroller

Our senior design group has chosen to utilize the MSP430 microcontroller family for this project subsequent to contrasting every one of the models that we compared above and numerous others. We thought long and hard about this but in the end this choice depended on various elements. Some elements being more important to us than others. Primarily, we require low power. A high power microcontroller would mean less space for other components and more weight to lug around. However, beyond that, we require low power utilization for three reasons: to augment the time we can have the KittyBot rolling around before waiting be revived and recharged. Furthermore, we have to minimize the measure of weight and size as mentioned earlier. To diminish the weight on the engines that will move our sphere and to minimize the dependencies of the sphere on large energy sources. To do this we needed to find a microcontroller that would facilitate this goal. Maintaining a strategic distance from a cumbersome battery would be a great design accomplishment for our team. A microcontroller that conveys a considerable measure of weight and requests a great deal of board space, is needless to say not what we want. We also wanted a microcontroller with sufficient peripherals and proficient I/O ports.

We did not start by considering which is the best microcontroller but rather which is the best microcontroller for our project. We researched far more than the microcontrollers referred to above and considered many advantages and disadvantages of devices. For example, we considered a PIC and thought that maybe it might be best for our group because of its minimum size. We found out they have incredibly small ones, but later saw to many complexities in implementation. If we needed a more potent processing power, on the other hand, we might have chosen to go with a cortex. Minimum power consumption, however, was an important aspect for us, hence the MSP430.

Another reason we have chosen the msp430G2 is the flexibility we have with the compilers. Texas Instruments provides a whole family of compilers to assist us in corralling a logical elegance of one and zero bits on the Printed Circuit Board. Texas Instruments Select the IDE you're comfortable with. To learn more about our software offerings such as Energia, CCS Cloud, and Code Composer Studio™

In order to achieve certain goals with our second robot design, we decided to implement the Arduino Uno microcontroller as well.

6.4 Sensors

6.4.1 GP2Y0A41SK0F

The Sharp GP2Y0A41SK0F analog distance sensor is a possible choice for the KittyBot to detect objects. The proximity sensor is especially desirable because they're cheap for photoelectric sensors and only cost 10 dollars. Considering the ear design calls for 2 of these sensors, they will save us more money than many of the other researched sensors. The range for this distance sensor can be set to a desired range of 4-30 centimeters.

The proximity sensors are low power which was wanted for design specification purposes. The proximity sensor requires a 4.5V - 5.5V input and the max current is 22 mA. The sensor draws current in short bursts and it's recommended to place a 10µF capacitor by the sensor across Vcc and ground in figure 6-1. The capacitor will stabilize the power supply from too large of a burst of current.

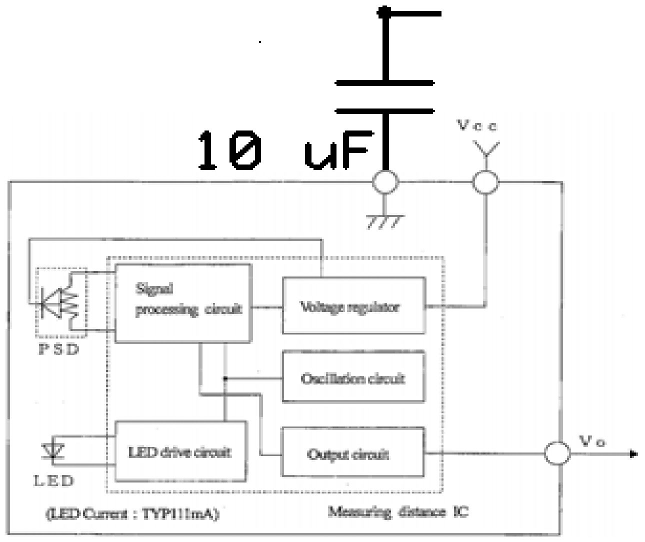


Figure 6-1

6.4.2 PING)))

The PING))) sensor from Parallax was heavily considered due to the groups familiarity working on other projects which included Parallax devices. It meets the requirements for the range as the sensor can work up to 3 meters. I found this sensor to work with the microcontroller being chosen, the MSP430. This works well with the KittyBot design because it is also 5V like the previous sensor and the motors. The PING))) sensor is simple to use considering the 3 pins and their functions. This sensor works using a 40 kHz signal. The frequency is burst out for 200 µs and can sense the object between 115 µs and t_{max} which is 18.5 ms, and has a 200 µs delay before another signal is sent out.

The PING))) ultrasonic sensor ultimately has some drawbacks that made the group come to a decision that it would be hard to do. First, it would be hard to implement with the design on the mobile KittyBot toy. If the design with ears were used, there would be large holes making the sensors vulnerable to the cat swiping in at the sensors and possibly causing damage. The other designs for KittyBot don't allow for these types of sensors since all the parts will be confined in a spherical casing.

6.4.3 Piezo Element

So after much research, we decided to use a Piezo element to detect any impact that the feline might inflict on our unit. Firstly, of course, we would have to figure out how to use the Piezo sensor. This sensor could be used for a large variety of reasons; be it to detect a knock on the door or the vibrations of a solid table. We are using to detect attacks from a cat. After researching the device we discovered that the piezo device is able to make a voltage after being physically altercated or irregularly touched. This could be due to a vibration from a physical element like a cat attack, a sound wave or any sort of a mechanical strain on the device.

When we say Piezo sensor, we are really using a short name for piezoelectric sensor. In essence, a piezoelectric sensor that measures changes. It has the ability to use the piezoelectric effect. This is basically a manipulation of the electric charge that might accumulate in solid materials. The piezoelectric sensor uses this effect to detect the changes in certain things. It can detect the changes in anything from temperature, strain, force pressure or electric charge. Where it is useful for us is in measuring the changes in charge and vibration. We will use this effect and the Piezoelectric sensor that leverages it to detect vibrations on the outside of our sphere. These will, of course, be applied due to the predatory felines attacks on our device.

This is a very useful tool for us because it captures the moment of interface between our unit and the predator feline. In addition to this, you can also place a voltage across a piezeo. If we do this the device will actually vibrate as well as create a tone of our liking. In essence our plan is to run a signal to an audio output when the kitty-bot is attacked. When rapid prototyping the unit we were able to scan and evaluate the output of the device using the analogRead() function provided by the library we are using. Next we had to encode the voltage. What we decided to do is to slice the voltages into different physical values.

We encoded the values to vary between the range of zero to five volts. In addition to this we assign an integer value that ranged from zero to one thousand and twenty-three. By splitting everything into these values we were able to apply an analog to digital conversion otherwise known as an ADC. Our goal was to be able to control when our sensor would react and when it wouldn't. First, as a proof of concept, we connected the entire unit to the computer. If the sensor was more powerful than a certain threshold our msp430 microcontroller would then send the command "React" to the computer. This would be done over the serial port. If we refer to the diagram we can see that one of the sensors' cords is connected to ground and the other to the microcontroller. The wire that's connected to the microcontroller is the serial port of the msp430.
Musical output added to our circuit.

Multiple Piezo sensors

As can be seen from figure **3.4.17** we at this point in Agile rapid prototyping sprint only have one sensor working. This one sensor, simply will not be sufficient for what we need. We need enough sensor coverage on the sphere where if any part of the our kitty bot is attacked we can take it as an input and react accordingly. Hence at this stage in our rapid prototyping process we decided it would be worth our while build multiple sensors into our core electrical design. What we came up with can be analyzed in the following figure.

Figure 6.4.18

Notice **Figure 6.4.18** it contains the final outline of our prototyping session with sensors. During our research process we realized that Piezo sensors are polarized. This sounds complicated but it basically means that any voltage that passes through their circuit will do so in one specific direction. This is in contrast to a bipolar sensor. In order for us to get many Piezos to work we found out that we must connect the black wires to ground in series and the red wires to the analog pins. However, we found out a quirk when getting in the other sensors. We must additionally to our previous design Also connect a one Mega Ohm resistor in parallel to each Piezo device. This serves a very important service to the totality of our device. It limits the voltage and current produce. The Piezo might not react well to the fluctuations we found so it was safer to do this. This also serves to protect the analog input from potential damage. We prefered to purchase the piezo sensors that looked like a metallic disc because it was less overhead and allowed us to deal directly with the sensor. Alos it doesn't have anything to impede it so it is easier to use as an input sensor. We also found other small but useful discoveries. If the Sensor is not firmly placed against the wall of our sphere it does not work as efficiently. In our final implementation we will build the sensor into our actual device for best production.

<u>Using our sensor to play musical notes</u>

When creating the project one of our main goals was to make it as fun as possible. After all we are making a toy for cats and humans. Throughout our process we began to find things that would help us actually implement this vision. One of the ideas we came up with was to make the unit so that it reacts with a unique or entertaining sound. No one can deny that we all like things that sound. For example a piano or a flute. But we dont like things that sound in an annoying way. For example a door bell. Well it turns out we can do this using the piezo and it works out great. In this way we can save by not putting extra components on our unit. As long as we have the tones and the durations of each tone we will be able to create what we want. Actually we can code the msp430 to play what we want.

For prototyping purposes we decided to have the sensor play happy birthday. To do so we went to a web site and grabed the notes necessary to create the melody which were as follows.

The notes were:

c c d c f e

c c d c g f

c c highc a bflat g

a a bflat f g f

The first time we did it we used the traditional delay to code what we wanted. We wanted a better way of doing this so later we figured out how to get the delay that we want using the PWM pule width modulation hardware. In order to get this to work we had to figure out that all notes correlate to a particular frequency. What we were able to do was to reset the channel duration for the second half and set the channel to use only half of the duration. Using other references, we were able to get a Pulse Width Modulation of fifty percent Duty cycle at our particular frequency. With the previously described set of logical steps we were able to accomplish something very impressive. We were able to make a square wave from the sine wave. After that, we, of course, encountered more problems. One of these being the ability to seize a Tone. We wanted to stop the Tone. The problem was that since, we were using PWM and software bit-banging we were now unable to simple zero out the output by zeroing the bit. That being said we also knew that we can make the period zero if we want. With all of this in mind, we coded the function and initialized our PWM pulse width modulation and were able to accomplish our goals. With this modification in place we will be able to accomplish a very nifty thing. Now every time the cat attacks our unit, the unit will sing a different musical note. When we consistently attack the unit it the unit creates a melody that we all can enjoy.

Sensor Selection

The initial plan for KittyBot's design included piezoelectric sensors. It was quickly realized that they would not be suitable to use. It would be impossible for the wires to stay untangled with the ball constantly rotating if the sensors were on the inside casing of the hamster ball. Therefore the KittyBot's hamster ball design didn't include a sensor. A second bot was created to show the ability to have a working sensor on a robot. A Parallax PING ultrasonic sensor was used for another KittyBot robot that would trigger a cat noise when it detected an object. The sound element was another component the team wanted show they were capable of adding to KittyBot for complexity. The ultrasonic sensor runs on 5 volts so it fits in with the power specifications well. The group decided to use an ultrasonic sensor over another similar sensor, a photoelectric sensor, because of the simplicity. The PING sensor is made from Parallax which is the same company that KittyBot's motors are from so the team was more familiar with the equipment. The GP2Y0A41SK0F photoelectric sensor was also highly considered because it was cheaper but the unfamiliarity with the product made us stray away from it. In the end the ultrasonic sensors worked with great success.

6.5 Printed Circuit Board (PCB) Design

A printed circuit board is a circuit board that electronically connects and mechanically supports electronic components using tracks, pads and other conductive features made from coppers sheets laminated onto a non-conductive substrate. The substrate is usually a semiconductor such as silicon, silicon dioxide or gallium arsenide that serves as a foundation upon which electronic devices like transistors, diodes and integrated circuits are deposited. Components are soldered onto the PCB in either a single sided, double sided or multi-layered layout. Our group decided that using a double-sided layout for our PCB would be most effective because it would allow us moderately high component density while keeping costs relatively low.

6.5.1 Layout and Design

For our PCB we decided to go with a basic double-layer design with mostly through-hole components to make soldering less difficult. Our overall design consists of a battery, a voltage regulator, three servo motors, a microcontroller and two sensors. The following figure shows our schematic which demonstrates how are circuit will be laid out on the PCB and how each component will be connected to each other.

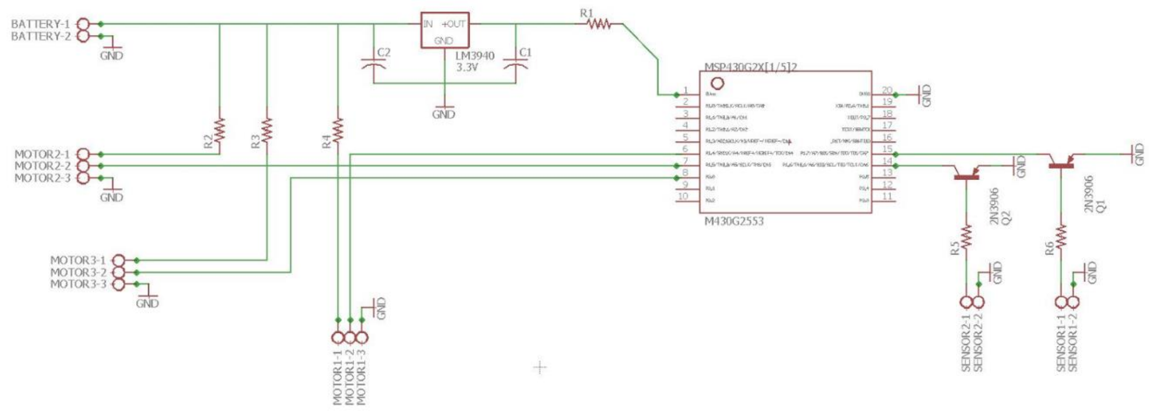


Figure 6.5.1: Schematic

We decided that ordering all the parts and soldering them on ourselves would be the best course of action. Due to this, it was imperative that we made sure there was enough room on the board for proper soldering because we are novices when it comes to that skill.

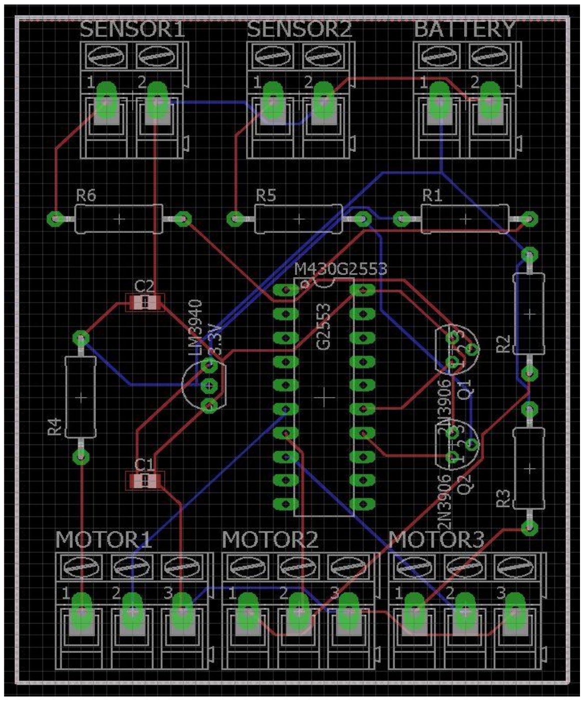


Figure 4.2: Board Layout

As you can see from the image above, in our board layout we tried to give our components as much room as possible while keeping our board size within our specifications. The largest board we could have that could fit in our chassis was 3in by 2.5in and we were able to make our board 2.95in by 2.2 inches.

6.5.2 Programming Microcontroller on PCB

During the prototyping and initial testing phases the project will be implemented using an MSP430F5529LP LaunchPad. This will provide the design team with a quick method of implementing the robot movement algorithm and interfacing the major components of the robot such as the MCU, Servos, Sensors, and Wireless Communication. One of the advantages of using the LaunchPad to prototype the project is that it has an emulator board on it that is used to program the microcontroller. The microcontroller cannot be programmed properly without this.

However, for the final design the project will obviously not be using a LaunchPad, but rather a custom designed PCB with the microcontroller and all other components surface mounted. The emulator board is not a simple set of components but rather a complex and high level emulator module that is beyond the scope of the design team. Also, the emulator board physically takes up a considerable amount of board space. The design team has made a decision to use the emulator board on the LaunchPad to program the MCU on the custom PCB for the final design. This seemed to be the most cost, time, and space efficient way to implement the design without having to redesign a Texas Instruments Emulation board.

To use the emulator board from the LaunchPad to program the custom PCB the design team must isolate the emulator board from the LaunchPad and replace the on board MCU with the MCU on the custom PCB. Instead of constructing the entire emulation board the design team will simply design the PCB so that jumper wires from the jumper block can be attached temporarily to set the MCU on the PCB as the target device for the emulation board. After considering different options and time constraints it seems that this will be the most efficient method to be able to use the emulator without having to reinvent it.

6.5.3 Soldering

While soldering may seem minor to the experienced engineer or hobbyist, none of the design team members have ever done it before. Since it is such a fundamental skill in electrical engineering and none of the team members have done it before, time will be taken here to gain some knowledge about how to solder.

Soldering is a process used to join different metal components together. This is accomplished by using a metal alloy (solder) to connect the different pieces by melting the solder onto the components and allowing it to cool. This creates a bond that is strong enough to hold the components together and also conduct electricity. Soldering is different from other methods used to fuse metals together such as welding because it occurs at a lower temperature (around 400 degrees Fahrenheit). Also, soldering melts a filler material between two metals to create contact unlike welding which actually melts the independent metals and fuses them together. Soldering can be “undone” for this reason by melting away the solder when it is desired to do so.

Soldering Tools

1. Soldering Iron

The size of the soldering iron depends on the application. A 15-40 watt soldering iron is good for circuit board soldering while 60-140 watt iron is better for thicker materials. Using a higher power iron on small components can result in overheating and damage to the components. Some soldering irons have variable temperature so that most applications can be accomplished with one iron however they are much more expensive.

2. Solder

Solder comes in a variety of thicknesses depending on what it is needed for. For circuit board applications thinner solder is better since it is more detailed work. Most solder material is combination of lead and tin but nowadays lead is being phased out of design due to health concerns. Some solder contains silver as well which results in a higher melting temperature which can result in burning components if care is not taken. Apparently solder with rosin core is better to use because it acts as a flux and helps the connection.

3. Soldering iron tips

Soldering irons come with tips but it is good to know what tips are better suited for certain applications. For detailed work, it is better to use a conical shaped tip while a flat larger tip is good for joining wires together. Also, the tip should be slightly smaller than whatever is being soldered.

4. Soldering iron holder and cleaning sponge

This just provides a safe place to hold the iron while not in use and a safe means of cleaning the tip.

5. Tools for wires and clips to hold work

Wire clippers and wire strippers for cutting and stripping wire. Also good clips to give extra hands while soldering pieces together are necessary such as “helping hands” or just alligator clips, anything to help make the soldering process easier.

6. Safety equipment

These include exhaust fans so that fumes are not being inhaled and safety goggles.

Soldering Procedure

1. Heat up soldering iron and clip all components together onto clips in the proper orientation such that the board can be flipped upside down and not have everything fall off.

2. Clean tip of soldering iron with a wet sponge.

3. (Soldering Wires Together) Strip about half an inch away from the ends of the wires and twist them together to form your joint. Touch the soldering iron to the joint (not the solder) and begin to heat the wires. Touch the solder to the wires (not the iron) and wait until it melts into the joint. If you touch the iron directly to the solder it will melt around and not into the wires and will form a “cold joint” and results in a poor connection.

4. (Soldering on a PCB) Place the leads of whatever component is needed through the hole in the PCB then bend it slightly so that it does fall out when flipped over. Touch the tip of the soldering iron to the led and metal pad on the PCB making sure that too much heat is not added that would damage anything. Once the lead and pad are hot touch the tip of the solder to the crack in paying careful attention to how much solder is applied. Too much solder can pool over connections and cause short circuits while not having enough can cause a poor connection. The right amount of solder will form an “ant hill” like mound. If this is not the case, make sure that all leads and pads are clean first. Remove the solder 1 or 2 seconds before the iron is removed so the tip of the solder does not stick to the connection; next cut off the excess lead as close to the PCB as possible with sharp wire cutters.

5. (Surface Mounting Components onto a PCB) The first step to surface mounting components onto a PCB is to “tinning” the pad. This is

accomplished by heating up the pad where you want to mount the component and applying a small amount of solder to it to create a small pool. Next you lower the component onto the solder and pad with tweezers and heat up the solder again to form the connection; hold the component in place for an additional few seconds to allow it to cool. Last, connect the other end of the component to the other pad by soldering the two contacts together.

6. (Desoldering and Fixing Mistakes) Desoldering is done using either a solder pump or desoldering braid. It is basically just reheating the joint, removing the solder and removing the component or resoldering the connection correctly. Fixing mistakes can be done by just reheating the connection and adjusting the component so that it is placed properly and has a good connection/ enough solder.

7 Software Design

When creating our project, it was necessary carefully analyze how to approach every aspect of our Software. Our project is an interwoven mesh of Electronics controlled by embedded systems. That being said our development environments were crucial to the outcome of our project. Essentially, the software team's job is to be the brains of the unit. We considered many frameworks along this process but came to solid reason based solutions to our processing needs. Firstly, it was necessary to consider what would be our Integrated Programming Environment. Many options were available, however, we came to the conclusion that it would be intelligent to come to a conclusion about what microcontroller we wanted to use before we decided on the Integrated Development Environment. After much deliberation, which was specified in detail in our section on microcontrollers we decided that the msp430 would be the ideal microcontroller for our purposes. We wanted a lightweight low energy high in community resource solution to meet our processing needs. The MSP430 was able to provide that. Additionally, it provided a plethora of integrated development environments to choose from.

7.1 IDE Options

The TI MSP430 line of microcontrollers is usually programmable through Texas Instruments' proprietary integrated development environment (IDE) Code Composer Studio. Code Composer Studio is a very robust IDE.

An alternative to using Code Composer Studio is Energia. Energia is an IDE for TI Launchpad microcontrollers that is very similar to the Arduino IDE. This will allow for us to use Arduino libraries.

A critical thought for this development is the way the hardware will be programed. Hence, we considered many Integrated development environments to fulfill our tasks. To efficiently integrate logic and intelligence into our hardware there must be a path for the equipment to be modified, tried and fixed as fast and productively as could be expected under the circumstances. In a perfect world the task would have been customized utilizing an environment that takes into account larger amount calculations to be actualized without focusing on controlling individual bits and registers. For example, MatLab might have a good consideration if we only needed to analyze data but unfortunately we needed to control hardware at its most fundamental level. To fulfill this, the software integration developers decided to use the Energia Integrated Development Environment. The choice to utilize the Energia Integrated Development Environment depended on various elements that were meticulously considered and evaluated.

In particular, one of the reasons that stand out amongst the most vital reasons our group picked Energia is that it is intended to use the Arduino programming algorithms and consolidates the plenty of libraries that can be actualized for any of the problems that we may encounter. This IDE has libraries committed to everything that our hardware programmed intelligence needs to execute. For example, Energia has a plethora of libraries that can drive engines dealing with the heartbeat of our PCB, that is the (PWM) Pulse width modulation required to do tasks. There are additionally libraries to peruse sensor information which will be unbelievably essential to facilitating the basic leadership process required to effectively execute the fundamental calculations we need. To efficiently navigate the labyrinths paths of development and expand the effectiveness with which they can be unraveled this integrated development environment will be highly useful.

Another helpful component that is connected with the Energia Integrated Development Environment is that coding representations can be transported in into Code Composer Studio which additionally accelerates the advancement of our compiler time. Our group can build up the essential calculation utilizing the Arduino framework and libraries at first. After that it can be transported and translated into the Code Composer environment. if more exact refinements are required, for example, advancement of register and control of individual bits, both calculations that are more difficult those can be done in Energia. Group five trusts that utilizing Energia in conjunction with Code Composer Studio will help expedite the aggregate of our teams programming and improvement abilities.

7.1.1 Potential IDE and our Choice of Energia

After careful consideration we came to the conclusion that the Energia Integrated Development Environment was best suited to meet our needs. One of the things we really liked about this development environment is that it is open sourced. This means that anybody can look at the code. Also, a community of people developed the environment so it is geared to real life practical needs. This software framework was exactly what we needed. The software framework is based on a Wiring framework and is capable of providing a non-technical easy to follow development workbench. When researching the codebase, we also realized that it is very robust and beyond that simple. Common sense is all that's really required in terms of training to use the Environment. The open source community provides a lot of benefits to its end users. Other IDE considerations are listed below. Though we were able to find a multitude of Integrated Development Environments that would could work conjointly with the msp430 we only truly considered the two considered below.

IAR Embedded Workbench:

Description	
IAR Embedded Workbench has a C and C++ compiler. It also has a debugger tool suite for applications. It can be used for MSP430 and TI ARM-based microcontrollers.	• Completely integrated development environment including a project manager, editor, build tools and debugger • Highly optimizing C and C++ compiler for ARM; Compatible with other ARM EABI compliant compilers. • Ready-made device configuration files, flash loaders and over 2800 example projects.

Mentor Graphics Sourcery Tools (formerly Code Sourcery, Inc.):

Description	
IAR Embedded Workbench is the world-leading C/C++ compiler and debugger tool suite for applications based on 8-, 16-, and 32-bit MCUs, including MSP430 and TI ARM-based microcontrollers.	• Completely integrated development environment including a project manager, editor, build tools and debugger

	● Highly optimizing C and C++ compiler for ARM; Compatible with other ARM EABI compliant compilers. ● Ready-made device configuration files, flash loaders and over 2800 example projects.

Though we considered the two previously mentioned Integrated Development environments they really didn't compare to Energia. We were astonished as to how easy to use Energia was. It doesn't have allot of options which is actually a good thing when you are getting used to a new Integrated Development Environment. Another amazing feature was the integrated Serial Monitor. This terminal extremely useful when testing sensors. We were able to see the TX and RX input and output real time. Also there are allot of API's that are plenty useful to us. These do advanced features with the sensors as well as helping us with controlling the microcontroller and peripherals. A feature which saved us countless numbers of hours we would have used developing low level elements. These included functions like digitalRead/Write and Serial.print amongst many others. We also found that this Integrated Development Environment was compatible with other devices we wanted to fiddle with. For example, the c2000 or the TM4C. Above all though we loved that the code was open source and hosted in the same GitHub server where we are holding our code where we can find higher level libraries for different applications we might use. If we ever needed a more professional environment we could also transition seamlessly into the Integrated Development Environment Code Composer Studio v6

7.2 Development Structure

7.2.1 Git Repositories over SVN

We decided to use Git repositories to maintain our code instead of Subversion. We came to the conclusion that git is better fit for our needs. We like the fact that Git is decentralized in its structure. With Subversion we can't have localized copies of our code. Also, with Subversion we might encounter a problem. We might be in a place where for example we might not have internet in which case we would have to literally copy and paste the code we would not be able to commit it. With git we don't have this problem our copy of the code is a local repository and we will be able to commit it whenever we please. That being said there is an added complexity to this approach. With git there is an entire language that we have to learn to track of our code. We also have to know the structure of the git system and how the branch structure works. Also we have to understand the difference between the local repository and hte actual branch that in our case would be kept on a server like github.

Git was at first a little confusing to us admittedly. We had to understand what it meant to work decentralized. What is a remote branch and also how to initialize and set up a repository? We were able to set up a centralized root branch of code. From that centralized root branch of code we set up development branches for each member on the software team. Each member now has his own branch to work on. When there is a change that the individual wants to keep he simply commits the change. The great thing is that this change still hasn't made it to the root branch. When the individual developer is ready to commit the changes to the main changes root branch he can do so. Then the administrator of the code can decide to pull the development branch into the root branch therefore updating the code. This as you could imagine is incredibly useful for our team.

7.2.2 Agile over Waterfall

When choosing our coding construct and framework of organization we had to think very carefully. From personal experience, if you don't properly map out your thoughts and plan the logical road map to success your logic will fail. Our main options for programing structures were agile and waterfall, we choose to use Agile. Some decades ago some programmers thought the waterfall methodology was not flexible enough to meet the needs of modern coding challenges. I happen to be of the opinion that they were right. That being said, many companies still use waterfall. They like the sequential and incremental approach of it. However, agile provides us more flexibility and potential for fast progress. We plan on starting off with a very simple algorithm design. Which we did in our prototypes. After that we began working on small portions of the code. We have organized ourselves for the work to be done on weekly and sometimes monthly sprints. When we complete the sprint we can then reevaluate the priorities of our coding approach. We then run our tests to ensure that our code is at optimum quality. The great thing about this approach is that this system helps us discover bugs as well as get feedback from our peers. This feedback can then be incorporated into the design and readjusted to create a better system for the next sprint. Some say that this is a very inappropriate approach because it lacks a serious initial design and sequential steps. However, the flexibility and creativity afforded to us by this approach suits our end goals perfectly.

In the end there were five main reasons why we chose the Agile approach over waterfall. Firstly, we need to be allowed to make changes to the code after the initial planning. This way we can rewrite things we found does not make sense. Secondly, because we are afforded the flexibility of making changes we will easily be able to add features to the code that will improve the excellence of our project. Thirdly at the finally of each sprint, we can then evaluate and reconsider our priorities. This will allow us to easily make adjustments to the project if our supervisor so desires. Fourthly, we believe strongly in testing and making excellence an integral part of our process. We need our code to run flawlessly when demoed. Failure could be catastrophic for our team. The infrastructure of Agile calls for testing to be complete at the end of each sprint. This ensures that the bugs that could derail our project are caught and preemptively disposed of at the end of each development cycle. Lastly, which is kind of tied to the previous advantage; because our code will be so thoroughly tested we could basically be ready to present working code at the end of any given month. This assures us and gives us security when crunch time comes around that worst case scenario we will be able to deliver a working product on demo day.

8 Project Prototype Testing

8.1 Rapid prototyping approach

The design phase of this project takes place during the UCF Summer semester of 2016. In order to improve our design, we put together a quick prototype during this time. We wanted to try and achieve a proof of concept on a motorized rolling ball to better understand the concept. Members of our group had a bo-bot complete with wheels and two servo motors from past personal projects. Our team also had a battery pack, breadboards, and multiple TI Launchpad MSP430 microcontrollers.

The first step was programming the MSP430 to rotate our servo motors. We achieved this with a simple program coded in Energia. The circuit was assembled on a breadboard and was powered by an external battery pack. We stacked the breadboard, microcontroller, and battery pack on the back of the bo-bot and turn it on. The bo-bot could successfully move. We then went about transferring this movement to a spherical object.

A hollow sphere that we could snap open and shut again was needed. In fitting with the pet theme of KittyBot, a hamster ball, 9 inches in diameter, was purchased. Our desire was to attach the two servo motors to the insides of the hamster ball so that the rotations of the servos could rotate the entire ball causing it to roll. We decided use the bo-bot's chassis in order to hold the servo motors. The bo-bot chassis was too large to fit in the hamster, so we sawed it in half. Figure 8-1 shows the internal components of KittyBot.

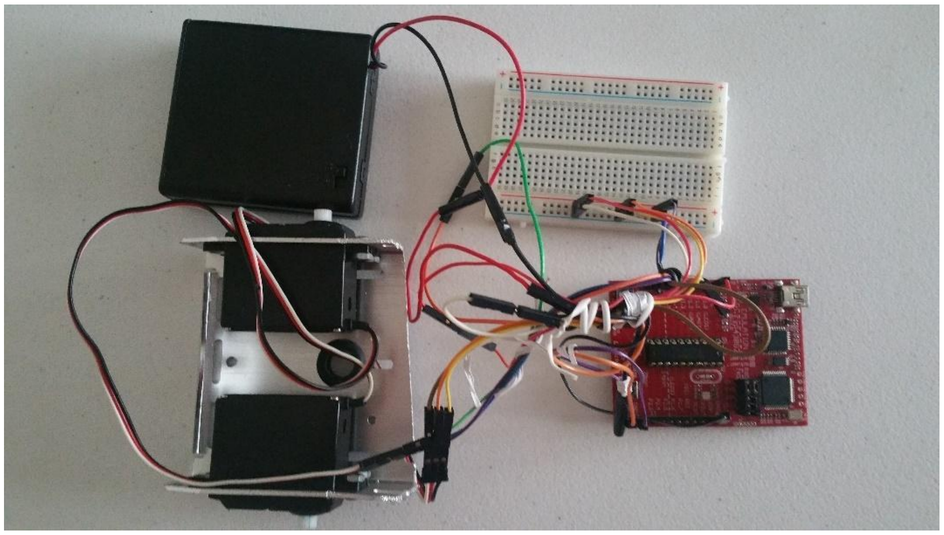


Figure 8-1 Internal Components of KittyBot

With the bo-bot chassis sawed, we had a good-sized housing for the two servo motors that could fit in the hamster ball. The chassis needed to be suspended in the center of the hamster ball with the servos attached to the opposite ends of the inside of the ball. Our quick solution to this was to drill holes into the bo-bots wheels and align them with holes drilled in the hamster ball. We would then insert a screw through the holes attaching the wheels to the hamster ball. A picture of the prototype at this stage is shown in Figure 8-2.

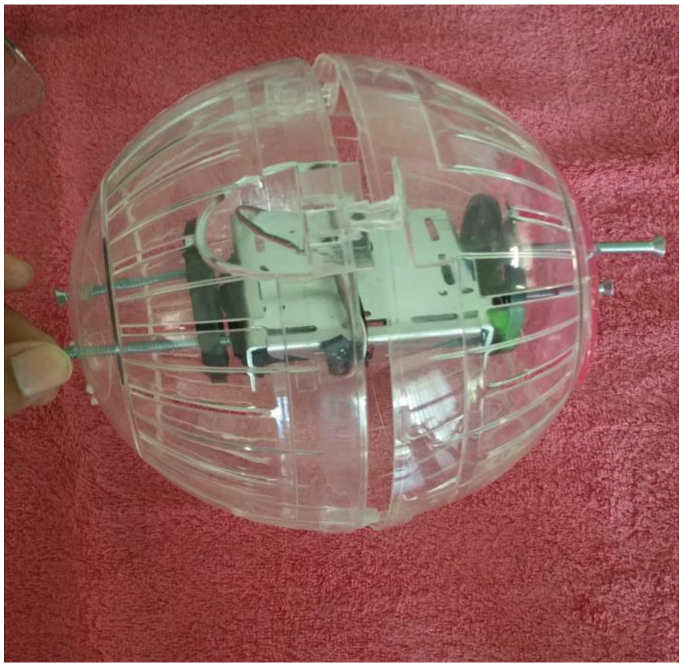


Figure 8-2: Chassis first attached to inside of hamster ball

With the chassis intact the microcontroller, breadboard, and battery pack needed to be placed inside the ball as well. The quick solution to firmly holding these components was rubberbands. In order to assembly the prototype for operation the code needed to be modified. The program was set to delay for one minute. This provided enough time to turn on the battery, reassemble the prototype, and place it on the floor in a ready position. After the minute delay, the servo motors kick in and rotate forward for another minute. This causes the ball to move. Finally, the servos are “detached” in the code, causing the system to stop. Below is a picture of the prototype at this stage (Figure 8-3).

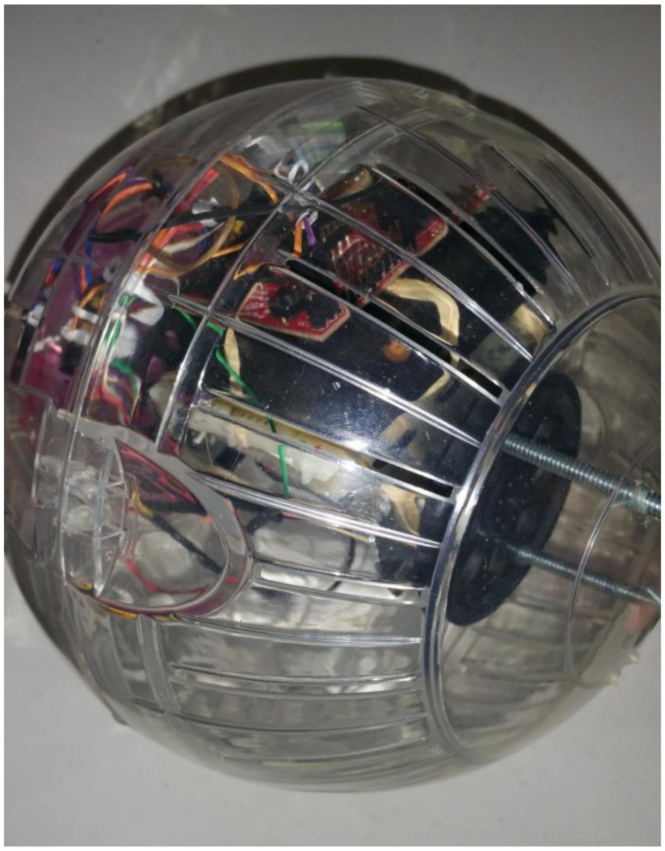


Figure 8-3 Prototype with electronics strapped to chassis

8.2 Design considerations derived from prototyping

Based on our prototyping approach, new design options needed to be considered. With the prototype displayed in Figure 8-3, the ball is able to roll due to the servos turning. Both servos turn forward, rotating the ball, however when the ball turns the center chassis ends up sloping forward. The components strapped to the top of the chassis combined with the torque of the wheels turning may cause the center unit to tip over. As the servos continuously turn, the center eventually tilts back upright just to move and fall back over again. Internally the center piece rocks back and forth. The instability of the central unit causes the movements of the entire system to be erratic. The system does not move straight or at a steady pace. When the central unit tips over, the prototype is halted because the servo motors' torque is moving the central unit back upright instead of rotating the ball forward. When the ball halts for that brief moment, it slumps over on one of its sides because the combined torque of the servo motors that keep the ball rolling on its vertical center axis stops momentarily. When the servos begin propelling the system forward again, the ball is starting from a leaning position causing the ball to veer off in the direction it was leaning. On one hand, this erratic, random movement is an interesting prospect for playing with cats. The random movement could potentially excite and entice the animals. Random movement patterns are desired for Kitty-Bot, but the group would rather achieve this through algorithms programmed into the microcontroller. The desired movement of Kitty-Bot is meant to be more controlled, because of this the physical instability of the prototype needed to be dealt with.

In order to stabilize the movement, the idea of a counterweight was introduced. Inspired by the schematic of the Rotundus GroundBot, the idea was to hang a weight under the central unit inside the ball. The weight should hang freely underneath and not connect to or touch the bottom of the inside of the ball. The free hanging weight should keep the central unit upright while the servos turn, allowing for straighter, more controlled movement. The central unit, consisting of the sawed bo-bot chassis, two servo motors, MSP430 microcontroller, breadboard circuit, and battery pack, was weighed; it measured 308g. A calibration weight weighing 500g (Figure 8-4) was tested first.

Figure 8-4 500g Calibration Weight

This weight was 2.5in tall. This made it worrisome as to whether it would fit inside the prototype without touching the bottom. Upon inspection, the weight was able to hang from the center chassis with just enough clearance. This weight however proved to be too heavy for the motors to handle. A valuable lesson was learned from this experiment. Weight is very important to consider. KittyBot needed to be as light as possible to avoid the need for larger motors. This would also keep the costs down and keep the overall size of KittyBot small enough to be acceptable as a household item.

With the limited supplies of this earlier stage we decided to test further levels of performance through programming. We decided to add turning to the prototype. By making the left wheel stop and rotating the right, the torque from the right wheel would cause the sphere to spin to the left. Pausing the left motor for about one to three seconds should give enough time to cause a 90 degree left turn. The inverse can be done to cause a right turn. We also programmed an about-face 180 degree turn and a reverse movement. To reverse the angles of spin of the motors are simply swapped causing the device to rolling in the opposite direction. Achieving 180 degree rotations are very similar to the 90 degree ones. The stopped motor just needs to be paused for longer and the active motor should run longer. Three to five seconds is enough to cause a complete about-face in the prototype. **Figure 8-5** and **Figure 8-6** show visual representations of the left and right rotations in 90 degree and 180 degree respectively. The views are from a top-down perspective. The yellow line indicates the forward facing direction of the prototype and the red arrows are the direction of rotation.

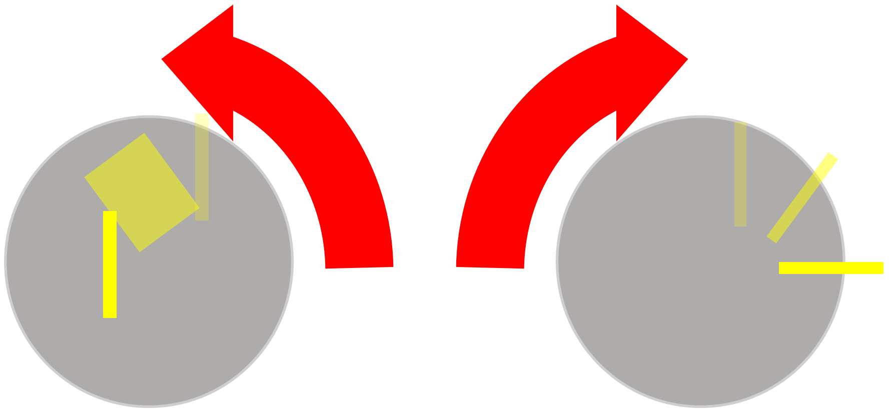


Figure 8-5: Visual Representation of 90 degree Left and Right Rotation

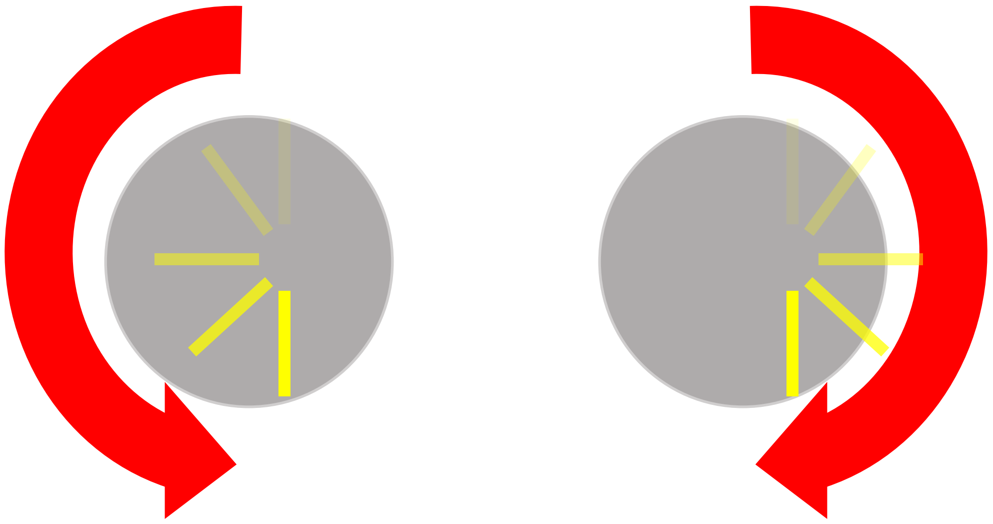


Figure 8-6: Visual Representation of 180 degree Left and Right Rotation

Figure 8-7 will provide a more detailed look at the movements.

Action	Turning Angle (in degrees)	Left Wheel Direction	Right Wheel Direction
Move Forward	0	Clockwise	Clockwise
Move Backward	0	Counterclockwise	Counterclockwise
Turn Left	90	Stopped	Clockwise
Turn Right	90	Clockwise	Stopped
About-face Clockwise	180	Clockwise	Stopped
About-face Counterclockwise	180	Stopped	Clockwise

Figure 8-7: Movement Details

8.3 Breadboarding

A good practice for testing and prototyping is to take notes of observations and data so that if a problem occurs, our team can go back and resolve it quickly. Using breadboard for prototyping allows the team the flexibility to swap resistors and other components in and out on the fly. It is smart to prototype this way because it allows the opportunity to make mistakes and learn from them before investing in a PCB. Of course, our final design will use a custom-made PCB tailored to the specifications of our project because it is more reliable and practical.

Before a resistor is placed on a breadboard, our team should ensure that the resistance value is determined by an online resistor calculator. If a component is a surface mount device (SMD) that has pins sticking out of its sides, then alligator clip shall be used to connect it with other components.

Using a multimeter and we can measure the amount of current drawn to the servos in the presence of no load and full load to understand how much the practical values deviate from datasheet values. We must make sure that the full load does not draw more current than the battery pack can supply. In case it does, we can replace it with another battery pack that can support higher current draw. The group will power the motors using a power supply in the lab before actually testing with a battery pack. It is best to set the current as low as possible then verify that the motors draw the right amount of current before slowly increasing the current.

Measuring the input and output voltage as well as the input and output current of the voltage regulator, we can then calculate the wasted power using Equation 8-1. If the power loss is within 1 watt, then no heat sink is required. If a linear regulator wastes more than 1 watt of power, then it should be replaced by a switching regulator.

Wasted power = (input voltage – output voltage) x output current (8-1)

Next, we calculate the voltage regulator's efficiency based on Equation 8-2,

Efficiency = (output power/input power) × 100 = [(output current × output voltage) / (input current × input voltage)] × 100 (8-2)

8.4 Conclusions reached

After observing the full movement capabilities of the programmed prototype, we decided we liked the movement. The movement was originally thought to be too unstable and erratic. The turning protocols programmed into the prototype provided an adequate degree of mobility. This dramatically helped us in determining if we wanted to pursue the spherical design fully. It is a design that can facilitate all of our requirements. The plastic outer shell is durable enough to withstand rough play from cats. It houses all the sensitive electronics inside behind a scratch resistant shell. Because it is a sphere even if it is turned over or tossed around it can still roll.

9 Administrative Content

9.1 Team Management

We developed a timeline for our periodic meetings throughout the semester once we decided on what wanted to do for our project. We generally try to meet up at least once every week to report to other members what we are currently working on, share some of the things we learn and new ideas that we may have. We also assign new tasks for the upcoming weeks. Our goal is to keep the project simple at first. As we research more, we'll add improvements to the robot and change our objectives if necessary.

The first two months is for research and prototyping. The last two months is for prototyping and testing as shown in Figure 10-1a. The research is very important so we knew we had to spend a lot of time on it. We have to write a report of 120 pages so each of us will write 30 pages. As we research, we write down what we learn in the report. We found out that writing everything down after all the research is done will take more time. The majority of the second semester of Senior Design will be spent purchasing components, designing the PCB and testing our algorithm. We know that it is important to order parts early and design a few working PCBs so that when the deadline approaches we can focus most of our attention testing our algorithm and troubleshooting. We are aware that system integration is important so time will be spent on that to make sure everything runs smoothly. When we created the table of contents, we also assigned areas of specialty to each member using a Divide and Conquer approach, as shown in Figure 10-2. We collaborated on some parts of the project as well.

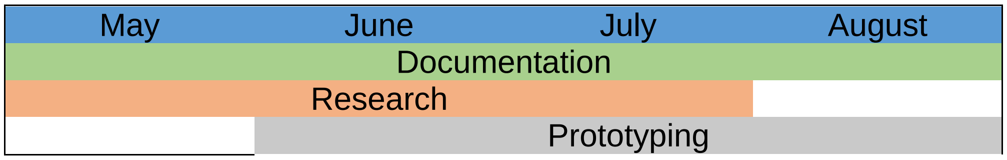


Figure 10-1a: Project Timeline for SD1

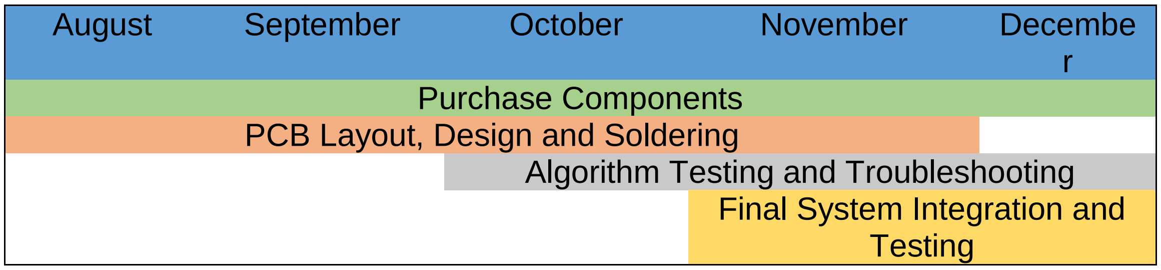


Figure 10-1b: Project Timeline for SD2

Divide and Conquer			
Bryen Buie	**Carlos Garzon**	**Stephen Barth**	**Trenton Williams**
Communication protocols	Microcontrollers	Motors	System Protection
System Integration	Movement Algorithms	Sensors	PCB Research
Data processing	Pseudocode	Communication Hardware	Schematic Design
PCB Design Circuit troubleshooting Code testing Error testing Bill of materials			

Figure 10-2: Divide and Conquer

9.2 Project Milestones

Most of us are pretty new to this project since we did not have any experience in robotics. Ideally, we aim to prototype and test in the second half of this semester. We overestimate the time needed to complete each task so that if an unexpected problem comes up, we still have enough time to fix. However, 75% of the time is already dedicated to research. We spend so much time on research because not all information we read about is related to what we are working on. Sometimes when we are building a part of the robot, we have to go back and research for more information.

While doing our own research, we have to keep in mind the other members' research too. For instance, if one member works on protection circuits and the other two are working on sensors and the microcontroller, the first member has to keep in mind of the specifications of the sensors and microcontroller. The research part is very crucial. It eventually determines how our product will turn out. One wrong step can lead to a series of problems in the future. Therefore, we pay very particular attention to our research. Initially, we wanted to finish the research part as soon as possible so that we have more time on the designing and prototyping. However, we are slightly behind schedule. Nevertheless, we all have solid understanding of where we are heading so we can be on track pretty quickly. If necessary, we'll spend more time on the project to speed things up. To increase productivity, we decide that each member should specialize in certain topics. This method saves time and prevents confusion due to overloaded information. Then we share what we learn with each other. However, we'll collaborate on the designing, testing, and coding because they are too important to leave to one member.

After we have finished a decent amount of research, we set out to acquire the components. We are looking for two factors: price and quality. For the sensors, the price is not too expensive so we should favor accuracy over price. For the microcontroller, we have to consider the price and the functionality such as the number of ports, the memory, and the processing power. Also, we need to know what kind of communication technology is compatible with that processor or launchpad. So far, we have acquired some of the material and just started with prototyping. Connecting the components together won't take too much time but getting them to work will take a lot of time. Interfacing the microcontroller with the sensors and motors are important to the robot's proper functionality. Therefore, this process will take a significant amount of time. We expect it to drag on for a few months. The milestone of the group for both semesters is as follows:

Senior Design I:

- Week of May 30 - Decide on Initial Project Idea
- Week of June 06 - Research sensors, microcontrollers, motors and other electronic parts.
- Week of June 13 - Design protective circuits & power supply (Hardware Team)
- Week of June 20 - Design protective casing/outer shell (Hardware Team)
- Week of June 27 - Design, simulate, & capture schematics (Software Team)
- Week of July 11 - Research and design algorithms (Software Team)
- Week of July 18 - Final Report
- Week of July 25 - Final Report
- Week of August 01 - Continue modifying and improving algorithm (Software Team)
- Week of August 08 - Continue modifying and improving algorithm (Software Team)

Senior Design II:

- Week of August 15 - Purchase hardware components (Hardware Team)
- Week of August 22 - Build chassis, connect motors (Hardware Team)
- Week of August 29 - Build pcb and other protective circuits (Hardware Team)
- Week of Sept 05 - Build protective casing and outer shell components (Hardware Team)
- Week of Sept 12 - Build power supply (Hardware Team)
- Week of Sept 19 - Interface components and test for proper connectivity (Software Team)
- Week of Sept 26 - Test sensors, collect and graph data (Software Team)
- Week of Oct 03 - Test and modify algorithm (Software Team)
- Week of Oct 10 - Test and modify algorithm (Software Team)
- Week of Oct 17 - Test and modify algorithm (Software Team)
- Week of Oct 24 - Build test area for kitten to play in.
- Week of Oct 31 - Test durability of play area and device with kitten
- Week of Nov 7 - Reinforce outer shell and play area if any weak spots are discovered (Hardware Team)
- Week of Nov 14 - Make sure the project meets expectations and is working as intended
- Week of Nov 21 - Make sure the project meets expectations and is working as intended

- Week of Nov 28 - Dec 5 - Improve and fix any problems or issues before presentation

9.3 Budget and Financing

As stated in the goals, the project's cost should be low, we estimate it should be around $200 but no more than $300. Texas Instruments' distributors Digi-key and Mouser have search filters that are simpler and easier to use than that of TI. Besides, the distributors allow buyers to select the mounting style which is not offered on Texas Instruments' website. The team will utilize the distributors' search filter to find TI products. We will also try to use websites such as Amazon.com with an Amazon Prime account and Ebay to order certain parts because we can get relatively fast shipping times and not have to wait two – three weeks for shipping when ordering from companies such as Digi-key.

Since there might be a chance that a component might become defective or break during testing and prototyping, electronic components will be purchased in multiple quantities. Nonetheless, the group should keep the quantity at a reasonable level, which is no more than 10, since it may be expensive to buy at a large amount. Besides, the team might not use the leftovers after Senior Design II. In order to save on shipping costs, our team will try to purchase only from a few sellers. Our team will first look for the items at local stores where we can pick up. If the items are not available locally, then we will look for them in online stores.

Four Rayovac rechargeable AA NiMH batteries may be purchased as a backup power supply besides the Energizer Lithium-ion batteries and charger combo which will be used primarily for testing. The Rayovac batteries work in all chargers so they can be charged using the Energizer charger, which is also advertised to be compatible with rechargeable AA and AA NiMH batteries. The Rayovac batteries are pre-charged so they are ready to use at any critical time when we need it.

Chassis is usually sold as a kit that includes wheels, motors, and battery holder as well as mounting parts. It may be more convenient and cheaper to purchase the kit than to buy the parts in the kit separately. However, a kit offers limited options. We have little choices in selecting the motors' size and type or dimensions and material of the chassis frame and wheels. Wheels can be purchased or acquired from old toys.

Figure 9-3 will list all the components needed for building the robot. The figure does not include the shipping fees. Therefore, the actual cost may exceed the total in the figure. All of the cost comes from the hardware. The software part is free. The list is subject to change in the future depending on the team's budget and when parts are ordered. Some materials may not be purchased if the team deems them unimportant to the success of the project or they can be substituted by another material.

Part	**Cost**
PCB	$50
Microcontroller	$40
Casing	$20
3D Print	$90
Proximity Sensors	$24
Power supply components and batteries	$30
Chassis to hold system	$10
Wheels	$20
Total	**$284**

Figure 9-3: Bill of Materials

10 Conclusion

The Kittybot consists of a robot that is designed to play with pets but it is mainly focused on kittens/cats. Group 5 has decided to work on a project that would not only challenge us, but also be useful and fun. Cats are often times curious and playful creatures. Their interactions with the robot would be entertaining for both the cats and their owners. The robot will be able to autonomously roam about an indoor space. It will also be able to sense its surroundings so it will not run into people, pets, or objects like walls, tables, couches, etc. Our goal is to create a robot that is small, cheap and power efficient.

Since the primary target for the robot is cats, it will be designed with this animal in mind. It will be of a small enough size (no more than 10 inches in height) as to be an appropriately-sized plaything for the average household cat. The robot will need to be durable enough to withstand rough contact from the animal. Cats have sharp claws and teeth, so the outer shell of Kitty-Bot must be scratch resistant, and the sensitive components such as microcontrollers, printed circuit boards (PCB), and wiring will need to be housed in durable compartments. Kitty-Bot may potentially be turned over while a cat is playing with it. If this happens, Kitty-Bot will be able to set itself upright again. This will be achieved by Kitty-Bot's spherical design. It will, in essence, be a "smart ball", an autonomous, self-rolling sphere.

Mobility is achieved through mechanisms of motors and autonomy is accomplished by the combination of custom codes designed by the team and pre-established software libraries. During Senior Design II, we expect to spend a lot of time-troubleshooting hardware problems and coding/software issues that may arise. Senior Design I is the research, design and prototyping phase. Since we are only prototyping in Senior Design I, some design areas will be lacking at first but will come full circle in Senior Design II.

This report is not meant to be followed strictly but to serve as a guideline for our design decisions and considerations when creating Kittybot. Adjustments and improvements will be made if a design, prototyping or testing is deemed inefficient, too costly or simply unfit for our goals. We have faced many obstacles in completing this report but we overcame them through our teamwork, perseverance and divide and conquer approach.

In the end, we succeeded in creating a system of robots that fulfill our requirements. **Figure 10.1**

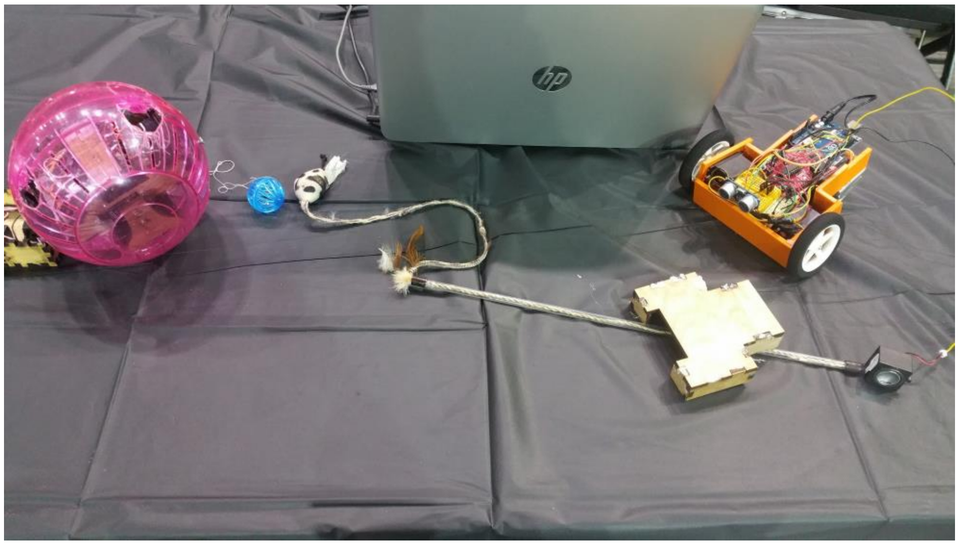


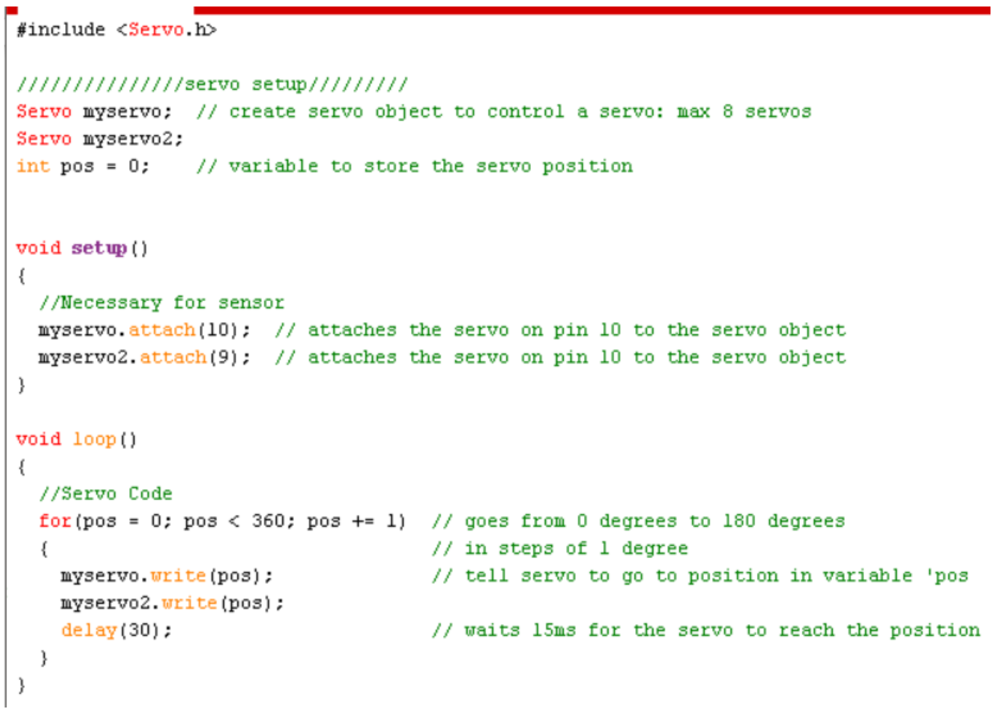
Figure 10.1 Final Project

Appendices

Appendix A - Copyright Permissions

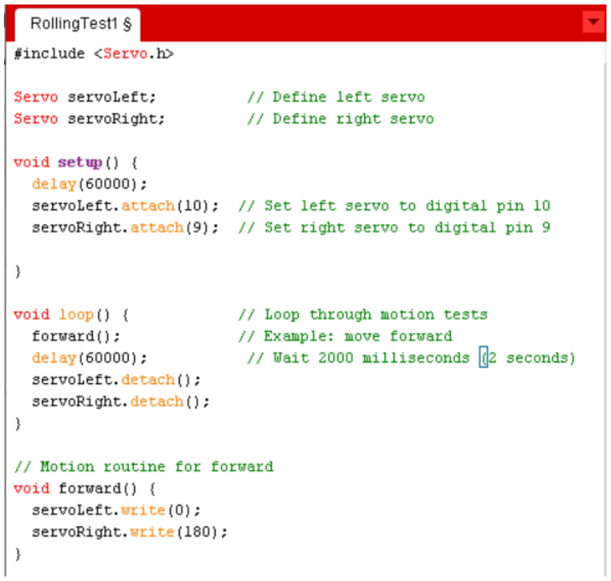
Sparkfun.com
nssn.org (standards)
Ti.com
Hexbug.com
Rotundus.se
Digikey.com
Anaren.com
https://www.arduino.cc/en/Tutorial/Knock
https://indiantinker.wordpress.com/2012/11/29/tone-library-for-msp430/

Appendix B – Code Snippets



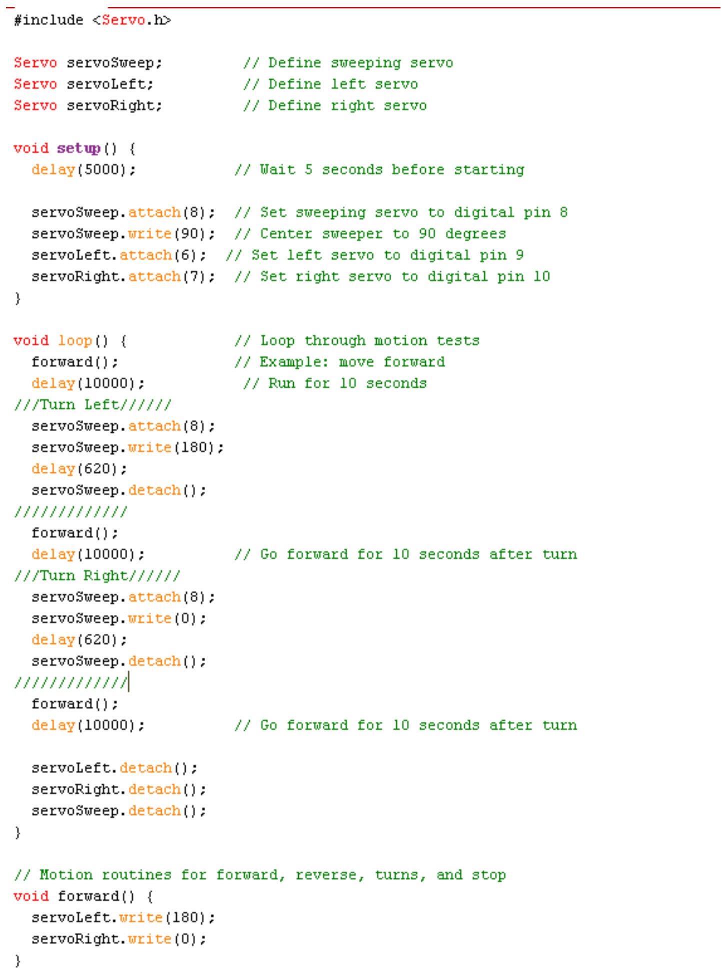
```
#include <Servo.h>

///////////////servo setup/////////
Servo myservo;  // create servo object to control a servo: max 8 servos
Servo myservo2;
int pos = 0;    // variable to store the servo position


void setup()
{
  //Necessary for sensor
  myservo.attach(10);  // attaches the servo on pin 10 to the servo object
  myservo2.attach(9);  // attaches the servo on pin 10 to the servo object
}

void loop()
{
  //Servo Code
  for(pos = 0; pos < 360; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos
    myservo2.write(pos);
    delay(30);                       // waits 15ms for the servo to reach the position
  }
}
```

Prototype Bo-Bot code snippet


RollingTest1 §

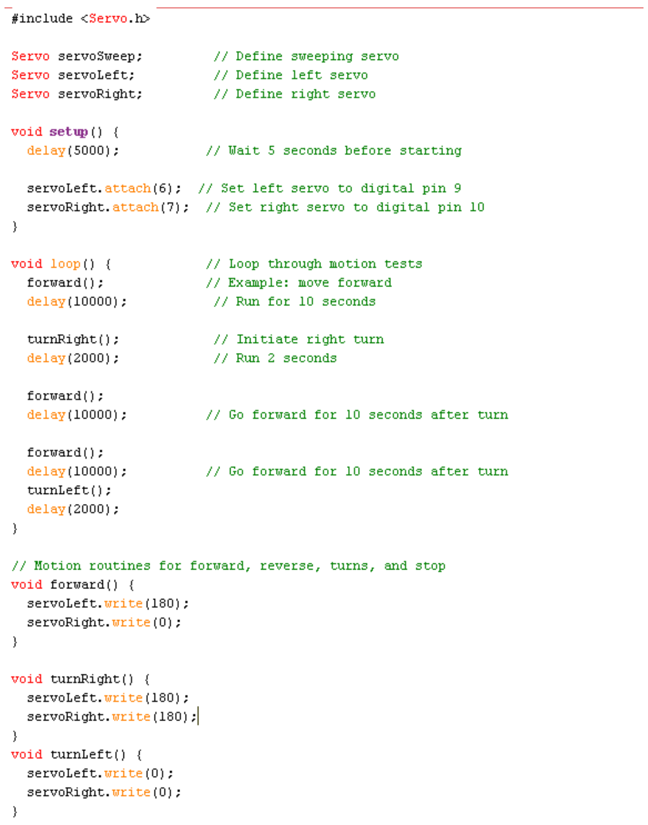
```
#include <Servo.h>

Servo servoLeft;          // Define left servo
Servo servoRight;         // Define right servo

void setup() {
  delay(60000);
  servoLeft.attach(10);  // Set left servo to digital pin 10
  servoRight.attach(9);  // Set right servo to digital pin 9

}

void loop() {            // Loop through motion tests
  forward();             // Example: move forward
  delay(60000);           // Wait 2000 milliseconds (2 seconds)
  servoLeft.detach();
  servoRight.detach();
}

// Motion routine for forward
void forward() {
  servoLeft.write(0);
  servoRight.write(180);
}
```

Prototype Code with Delay



```
#include <Servo.h>

Servo servoSweep;         // Define sweeping servo
Servo servoLeft;          // Define left servo
Servo servoRight;         // Define right servo

void setup() {
  delay(5000);           // Wait 5 seconds before starting

  servoSweep.attach(8);  // Set sweeping servo to digital pin 8
  servoSweep.write(90);  // Center sweeper to 90 degrees
  servoLeft.attach(6);  // Set left servo to digital pin 9
  servoRight.attach(7);  // Set right servo to digital pin 10
}

void loop() {            // Loop through motion tests
  forward();             // Example: move forward
  delay(10000);           // Run for 10 seconds
///Turn Left//////
  servoSweep.attach(8);
  servoSweep.write(180);
  delay(620);
  servoSweep.detach();
/////////////
  forward();
  delay(10000);          // Go forward for 10 seconds after turn
///Turn Right//////
  servoSweep.attach(8);
  servoSweep.write(0);
  delay(620);
  servoSweep.detach();
/////////////
  forward();
  delay(10000);          // Go forward for 10 seconds after turn

  servoLeft.detach();
  servoRight.detach();
  servoSweep.detach();
}

// Motion routines for forward, reverse, turns, and stop
void forward() {
  servoLeft.write(180);
  servoRight.write(0);
}
```

KittyBot Sphere Code



```
#include <Servo.h>

Servo servoSweep;         // Define sweeping servo
Servo servoLeft;          // Define left servo
Servo servoRight;         // Define right servo

void setup() {
  delay(5000);           // Wait 5 seconds before starting

  servoLeft.attach(6);  // Set left servo to digital pin 9
  servoRight.attach(7);  // Set right servo to digital pin 10
}

void loop() {            // Loop through motion tests
  forward();             // Example: move forward
  delay(10000);           // Run for 10 seconds

  turnRight();            // Initiate right turn
  delay(2000);            // Run 2 seconds

  forward();
  delay(10000);          // Go forward for 10 seconds after turn

  forward();
  delay(10000);          // Go forward for 10 seconds after turn
  turnLeft();
  delay(2000);
}

// Motion routines for forward, reverse, turns, and stop
void forward() {
  servoLeft.write(180);
  servoRight.write(0);
}

void turnRight() {
  servoLeft.write(180);
  servoRight.write(180);
}
void turnLeft() {
  servoLeft.write(0);
  servoRight.write(0);
}
```

Sensor Robot Movement Code