Indiana University-Purdue University Fort Wayne
Department of Engineering

ENGR 410 - ENGR 411
Capstone Senior Design Project

Report #2

Project Title: Team of Football Playing Robots

Team Members: Armela Mane (ECE)
Raihan Mir (ME)
Jeremy Nyikos (ME)
Cliff Sidwell (ECE)
Matthew Thompson (ME)
Colton Witte (ECE)

Faculty Advisors: Dr. Zhuming Bi (ME)
Dr. Guoping Wang (ECE)

Date: May 6, 2013
Table of Contents

Acknowledgments .......................................................................................................................... 4
Abstract ........................................................................................................................................ 6

Section I: Design Description ....................................................................................................... 8
  1. Problem Statement .................................................................................................................. 9
     1.1 Requirements and Specifications ....................................................................................... 9
     1.2 Given Parameters .............................................................................................................. 9
     1.3 Design Variables ............................................................................................................. 10
        1.3.1 Hardware .................................................................................................................. 10
        1.3.2 Software ................................................................................................................. 10
  2. Detailed Conceptual Design .................................................................................................... 11
     2.1 Hardware .......................................................................................................................... 11
        2.1.1 Microcontroller ........................................................................................................ 11
        2.1.2 Handheld Controller ............................................................................................... 13
        2.1.3 Motor Controller .................................................................................................... 13
        2.1.4 Accelerometer Kit ................................................................................................. 14
        2.1.5 Charging System .................................................................................................... 14
     2.2 Software ........................................................................................................................... 14
        2.2.1 Circuit Diagram ....................................................................................................... 14
        2.2.2 Software Flow Diagram ......................................................................................... 16
     2.3 Systems .............................................................................................................................. 17
        2.3.1 Locomotion ............................................................................................................. 17
        2.3.2 Center to Quarterback Ball Transfer ......................................................................... 17
        2.3.3 Quarterback Ball Transfer ...................................................................................... 19
        2.3.4 Netting System ....................................................................................................... 19

Section II: Building Process ......................................................................................................... 20
  1. Description of Building Process ............................................................................................ 21
     1.1 Center and Receiver Baseplate and Chassis ................................................................. 21
     1.2 Netting System for Receiver ......................................................................................... 23
     1.3 Center to Quarterback Ball Transfer .............................................................................. 24
     1.4 Quarterback Baseplate and Chassis .............................................................................. 29
  2. Description of Software-Programming Process ..................................................................... 31
Acknowledgments
The project would not have been possible without the help and support of many people. A special appreciation to the project advisors Dr. Zhuming Bi and Dr. Guoping Wang who provided knowledge and expertise which have been proven to be an invaluable help. Sincere gratitude goes to the University of Notre Dame for sponsoring this project and IEEE for granting supplementary funds.

We thank Notre Dame Professors, Dr. Michael Stanisic and Dr. James Schmeidler, for taking time and coming to IPFW to give us a background of robot football and a solid foundation to start upon. We also thank Notre Dame student Joe Rudy for helping throughout this process by answering our multiple questions and concerns.

A special thank you goes to SolidWorks Company for providing a one year software license to each team member for sketching the mechanical drawings. Finally, an immense gratitude and thanks goes to Mr. Jan Witte and Mrs. Betty Witte for providing us with a location and mechanical/electrical tools necessary to construct the robots; without their generosity we would have not being able to successfully complete our design.
Abstract
Robotics technology holds the potential to transform the future of the country and is likely to become as ubiquitous over the next few decades as computing technology is today [1]. To accelerate innovation in robotics, the Notre Dame University plans to create an intercollegiate mechatronic football league; the teams in the league compete against each other in robot football games. The first game was successfully organized and reported as a featured story by many influential media such as USA Today and NFL [1, 2]. The next step for Notre Dame University is to promote this league to a national level. Our senior design group has been selected as a new robot football team sponsored by the organizer.

The goal of this project is to build a football-playing robot team that will be competing in the Intercollegiate Mechatronic Football game at University of Notre Dame in April, 2013. Due to time constrain and limited team size, only three different robots of a complete robotic team are built. The selected robots are the quarterback, receiver, and center.

The robots’ design has multiple requirements that were definite in the Rules and Regulations of Collegiate Mechatronic Football and others were gathered from different prospective. One of the most important requirements specifies each player’s weight and dimensions. In fact, players must fit within a 16x16x24 inch box, and cannot weigh more than 30 pounds with an exception for the quarterback who can weigh up to 45 pounds. If these conditions are not met, the project would be a failure. In order for the project to be successful, the robots must be able to travel at a high speed, pass the ball from center to quarterback reliably, and pass the ball from quarterback to receiver reliably as well. The quantitative numbers that the team put together as goals for the project are a delivering operation from center to quarterback within 20 seconds with a success rate of 75%, success rate of 65% for complete passes, ability of receiver to travel 50 feet within 5 seconds, and ability of center to travel 50 feet within 8 seconds. The final requirement and constrain of the project is cost. The sponsor has given a budget of $5,000 and IEEE has given a grant of $500 for a total of $5,500.

The hardware utilized to construct the robots and an explanation of the software implementation is specified in detail. A brief narrative of the building process is included listing all the difficulties and problems that were resolved by modifying the original conceptual design. Testing of each player is conducted and evaluated to acknowledge how well the final design meets all set specifications and goals. Finally, evaluation and recommendations are incorporated to assist and advise prospects students to enhance the robot design.
Section I
1. PROBLEM STATEMENT

Robots have an ever-growing influence on our daily lives. Robots are a familiar example of mechatronic systems and are ideal student project due to the necessary application of Science, Technology, Engineering, and Mathematics (STEM) knowledge in creative engineering designs. To accelerate innovation in robotics, University of Notre Dame is creating an intercollegiate mechatronic American football league where robotic teams from the participating schools compete against each other.

Our senior design group has been selected to kick-off a new robot football team for IPFW, which will be sponsored by Notre Dame. Due to the limitation of having just six team members, only three robots (also referred as players) will be expected to be built during senior design project. The three players to be completed are the quarterback, receiver, and center, and they will be built using IEEE standards including those for wireless, software, and system engineering[3]. These three positions are the most critical part to the execution of a football game and also provide the most challenging design opportunities. The completed players will compete in the Collegiate Mechatronic Football game in April, 2013.

1.1 Requirements and Specifications

A complete football robot team consists of eight players on the field. Robots can be divided into offensive and defensive type roles. The members of the IPFW team will focus on offensive players including the quarterback, center, and a receiver, while Notre Dame will provide the defensive players from previous years.

- Robots must be capable of travelling 50 feet within 5 seconds starting from rest
- Robots must be equipped with digital accelerometer to sense upsetting event
  - Signal lighting system for 2 seconds
  - Remove power to drivetrain for 2 seconds
- Be able to replace batteries within a minute
- Center must deliver the ball to quarterback with a success rate of 75% within 20 seconds
- Quarterback must complete a pass 65% of the time
- Receiver must be able to travel 50 feet within 5 seconds
- Center must be able to travel 50 feet within 8 seconds

1.2 Given Parameters

The following given parameters are specified by the organizer to ensure fairness in the contest; they are explained in more detail in “The rules of collegiate mechatronic football” Report submitted by Notre Dame [5].

- Players must be DC powered, with a 24 volt maximum circuit voltage.
- Must include an externally mounted kill switch to stop power to drive train for security reason
- All robots must fit within a 16x16x24 inch box
- The centerline must be located 3.0±0.1 inches above the playing surface
- Material used must be High Density Polyethylene (HDPE) with a bumper of extruded Ethylene Propylene Rubber (EPDM) that covers perimeter of base plate
- All robots are remotely controlled from the field
- Receiver and center cannot weigh more than 30 pounds, while quarterback cannot weigh more than 45 pounds
- Cost of all robots is limited to $5,500

1.3 Design Variables
The design variables include hardware and operating conditions that may be varied to achieve the requirements of the total system as outlines earlier. Some parameters are practically unconstrained with others must fall within a specified range.

1.3.1 Hardware
- Base Plate shape
- Location of drive and not-drive wheels
- Motor and Gearbox
- Netting System for receiver
- Ball Transfer System for Center and Quarterback
- Handheld Controllers
- Tracking Hardware

1.3.2 Software
- Microcontroller
- Motor controller
- Programming Language
- Communication network and standards
2. DETAILED CONCEPTUAL DESIGN
The conceptual design is divided in three different sections: hardware, software, and systems. A detailed description of each subsystem and their components is given below.

2.1 Hardware

2.1.1 Microcontroller
The robots use Arduino Due microcontrollers. The Due board is a better fit for the purpose of this project by offering more advanced capabilities and functionality.

The Arduino Due, as seen in Figure 1, is based on the Atmel SAM3X8E ARM Cortex-M3 CPU and it is based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins, 12 analog inputs, 4 UARTs, an 84 MHz clock, a USB OTG capable connection, 2 DAC, 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button. The microcontroller operates at 3.3 V and is powered by a recommended input of 7V to 12V from an external power source.

The board is programmed with the Arduino 1.5.2 software. The libraries are written in C and used in the Arduino IDE. The program is uploaded through the Programming port using a Micro-B USB cable.

2.4 GHz Xbee module from Digi is used for wireless communication between remote controllers and microcontrollers. Zigbee communication protocol, based on an IEEE 802 standard[^4], is implemented for its low-cost and capabilities. The Xbee module is programmed using a USB to serial base unit as seen in Figure 2. This unit allows for configuration of the Xbee modules via X-CTU software.

Figure 1: Arduino Due Microcontroller

[^4]: IEEE 802.15.4 Standard for Wireless Personal Area Networks (WPANs)
A MEGA prototype board is mounted on top of the Due for clean and secure wiring. Although the shield is intended to be used on an Arduino Mega board, it is compatible with the Due. The Xbee module is held on top of a shield which is soldiered on the prototype board. A view of the entire microcontroller assembly can be seen in Figure 3. The wiring connections between the Due and the Xbee are run through and under the prototype board as seen in Figure 4.
2.1.2 Handheld Controller
The robots are remotely controlled using Arbotix Commander V2.0. The remote communicates with the microcontroller through a mounted Xbee module on its front plate as seen in Figure 5. The Commander is powered by 4 AA batteries, two on each side. The battery sockets are secured on the back plate using Velcro. This modification is implemented to speed up timing necessary to replace batteries as the sockets are intended to be held with screws. The VCC switch must be set to 3.3V when operating controllers that are powered with batteries. The M_RST_AUTO switch must be set to M for allowing manual reset.

![Arbotix Commander V2.0](image)

**Figure 5: Arbotix Commander V2.0**

2.1.3 Motor Controller
The robots use Sabertooth 2x25 motor controllers as seen in Figure 6. Each motor controller can regulate 2 motors at most. The Sabertooth is recommended to be powered by 5 to 18 cells high capacity NiMH or NiCd. The motor controllers operate using Packetized serial mode which uses TTL level multi-byte serial commands to set the speed and directions of the motor. Because packetized serial is a one-direction only interface, multiple Sabertooth can be connected to the same serial transmitter. The target device is selected using an address byte that is set via the dip switch; up to 8 Sabertooth (128 to 135 address byte) can be ganged together on a single serial line.

![Sabertooth 2x25 V2.0](image)

**Figure 6: Sabertooth 2x25 V2.0**
2.1.4 Accelerometer Kit
Notre Dame provided each robot with an accelerometer kit to detect a tackle event. The kit includes a custom made accelerometer, an LED light holder, and a power jack cable. The accelerometer comes already preconfigured and does not require any set up or additional changes.

2.1.5 Charging System
All the robots use a 7.2V 4500mAh NiMH battery to power up the Arduino Due and the Accelerometer. The Center uses the same battery to power the servo as well. The Receiver and Center motor controllers are powered by 12V 10Ah NiMH battery while the Quarterback uses 12V 20Ah NiMH battery. The handheld controllers require 4 AA batteries each.

2.2 Software
The microcontroller code is slightly different for each player due to their different capabilities and functionalities. The program is developed using Sabertooth and Commander Libraries that can be downloaded and added to the Arduino IDE following the detailed instructions included in the Appendix.

2.2.1 Circuit Diagram
The circuit diagrams for the Receiver, Center, and Quarterback are shown in Figure 7, Figure 8, and Figure 9 respectively.

The receiver has the simplest diagram; it has 2 drive wheels controlled by one Sabertooth 2x25. The motor controller is powered by a 12V battery that’s in line with a kill switch. When the kill switch is engaged, power is removed to the drive train. A GND wire and a signal/communication wire (TXO) are the only connections to the Due board. The accelerometer and the microcontroller are powered by a 7.2V battery. The accelerometer is connected to the status LED and to the Due through a power jack cable that connects to GND and PIN13. The Xbee connects to microcontroller 3.3V, GND, TX, and RX signal pins.

The center circuit diagram is equal to the receiver with the addition of a servo to operate the passing mechanism. The servo is powered by the 7.2V battery and connects to the microcontroller through PIN12.

The quarterback circuit diagram is a complicated version of the receiver circuit diagram. In fact, the quarterback uses 4 motor controllers: 2 for the four Omni-wheels, 1 for the two pitching wheels, and 1 for the lead screw mechanism. All motors are powered by the same 12V battery and connect to the microcontroller through a single GND and signal (TX0) wire.
**Figure 7:** Receiver Circuit Diagram

**Figure 8:** Center Circuit Diagram
2.2.2 Software Flow Diagram

The program flow, shown in Figure 10, starts by initializing ports to be inputs, outputs, variables and hardware interrupts. If the accelerometer signals the microcontroller this is known as “tackle”; when a tackle happens, the robot stops power to the motors for 2 seconds and the accelerometer updates the robot status LED. When the robot is not tackled, the robot reads the data from the UART; if no signal is being received the robot does not powers the wheels. If the signal is good the robot updates the individual motors speeds and performs that individual robot function. In the case of omnidirectional wheels, each wheel is governed by a unique equation that allows the Quarterback to travel in any direction and be able to turn about its central axis when desired. The Quarterback remote controller provides the signal for the pitching wheels speed. The remote also provides the signal to pass the ball which activates the motor on the lead screw and advances the ball into the pitching wheels. Then the program repeats the entire process continuously. Because both the Center and Receiver perform much easier operations than the Quarterback, their program flow is not covered in detail.

Figure 9: Quarterback Circuit Diagram
2.3 Systems

2.3.1 Locomotion
The drive trains of the Center and wider Receiver both use two driven wheels located in the center of the robots with 2 ball casters, one located in the front, and one located in the back, for stability. The Quarterback uses four omnidirectional wheels to move. This configuration also requires four motors and gearboxes. All of the hardware used in this omnidirectional configuration except the actual wheels is the same as what is used on the Center and Receiver.

2.3.2 Center to Quarterback Ball Transfer
The center, as illustrated in Figure 11, has the responsibility of placing the ball accurately into the quarterback’s ball feeder. This requires precise alignment with the quarterback in the plane of the playing field with respect to both translation and orientation of either robot.
This alignment is accomplished using the trapezoidal cutout section on the rear end of the center, which mates with the complementing male end on the quarterback. A four bar linkage arm system, driven by a servo is used to transfer the ball. As seen in Figure 12, the gripper uses tension spring to release the ball into the Quarterback’s ball feeder.
2.3.3 Quarterback Ball Transfer
The Quarterback top plate consists of a feeder mounted at a 35° angle. The ball holder is driven by a lead screw with a servo attached to it. Two wheels are used for the pitching mechanism; the wheels are the same as the ones used for the drivetrain of Center and Receiver.

Figure 13 shows a representation of the base plate of the Quarterback. The base is shown with the Omni-wheel drive train design.

![Figure 13: Quarterback Base](image)

2.3.4 Netting System
The Receiver uses cross pattern netting as shown in Figure 14. The netting system is attached to the lid of the Receivers’ chassis. 5 PVC pipe poles are attached to the lid to support the nets. In order to retain the ball, a cage is included on the perimeter of the lid of the chassis.

![Figure 14: Receiver Netting System](image)
Section II
1. DESCRIPTION OF BUILDING PROCESS

1.1 Center and Receiver Baseplate and Chassis

The first stage of the building process was to build the chassis of the Center and Receiver. The baseplates are made of 0.5 inch thick High Density Polyethylene (HDPE) while the side panels and lids are made of 0.25 inch thick HDPE.

First, the HDPE pieces for baseplates, lids and side panels were cut down from a large sheet of HDPE using handsaw as seen in Figure 15. The Receiver baseplate was cut to dimension of a 15x15 inch square. Slots for 4 side panels inside the baseplate were created using handsaw and hand drill. These slots are 0.5 inch inside from the edge of the baseplate. Similarly, slots for the wheels and motors were created.

![Figure 15: (a) Center baseplate with drivetrain spacing and (b) Side Panel of the chassis](image)

Next, the wheels attached to gearboxes and motors were mounted to the baseplate. Initially, the motors were mounted using manually manufactured mounting brackets made of 0.5 inch HDPE as shown in Figure 16(a). However, alignments of the wheels were found to be an issue. To fix the alignment issue, the wheels were mounted to a 0.25 inch HDPE piece as seen in Figure 16(b). This plate allowed us to align the wheels in parallel without further damaging the baseplate; the HDPE piece is mounted to the baseplate using bolts and nuts. Washers are put in between the HDPE plate and motor to maintain the baseplate height at 3 inches from the ground.
Figure 16: (a) Wheels mounted with bracket, (b) Wheels mounted with plate to fix alignment

The front and back of the baseplate is supported by 2 ball caster as seen in Figure 17. Each ball caster is mounted to the baseplate using two 0.5 inches thick 3x3 HDPE pieces and 2 inch long bolts and nuts.

Figure 17: (a) SolidWork model of the drivetrain, (b) Bottom view of the built drivetrain

The side panels are attached to each other using L-brackets and non-locking polypropylene draw latches are used on two sides to secure the lid on top of the side panels. A complete view of the Receiver chassis is show in Figure 18 below.
1.2 Netting System for Receiver

The cross pattern netting system of the receiver is installed using five 15 inches long 0.84 inch diameter PVC pipes as seen in Figure 19. Four poles are installed in four corners while the fifth pole is installed in the center of the chassis. Each pole is secured to the lid using two 0.5 inches thick 2x2 inches square HDPE plates. The maximum height of the poles from the top of the lid is 13 inches.

![Figure 18: Receiver assembled chassis](image)

The nets are attached to the poles using cable ties (zip ties). A detailed view of the netting is show in Figure 20. Strength cargo netting is used rather than nylon sport netting of 150 lbs. This modification
was applied to increase the probability of retaining the ball. To install the nets, 4 pieces of 0.25 inch thick HDPE are attached in between the center pole and corner poles.

![Top-view of Cross-pattern netting system](image)

**Figure 20**: Top-view of Cross-pattern netting system

### 1.3 Center to Quarterback Ball Transfer

The Center has the duty of placing the ball accurately into the quarterback’s ball feeder at the start of every play. For precise passing capabilities, a repeatable alignment is accomplished using the following design.

The base of either robot was designed to align with the other by using a trapezoidal notch cut out from the Center’s base and a complementing protrusion cut from the Quarterback’s base. A SolidWork view of each robot base is shown in Figure 21. On the other hand, Figure 22 shows the actual base built for each robot.

![Center and Quarterback base in SolidWork](image)

**Figure 21**: Center and Quarterback base in SolidWork
Once the two robots are aligned with one another at the line of scrimmage, the Center must lower the ball and release it to the Quarterback. This is done using a four bar linkage that is driven by servo mounted off the base plate. Originally, this process used a design that incorporated a wedge on the ball holder of the Quarterback that would pry open the gripper on the Center, causing the ball to release exactly as the gripper reached the ball holder. Figure 22 and Figure 23 below display this passing mechanism. This design was intended to eliminate fumbles or sloppy handoffs that could be caused by dropping the ball above the quarterback’s ball holder.
One major flaw existed with this design though, and that is that the Center would only be able to release the ball to the Quarterback we designed. It could not release the ball to any other robot, such as a running back or another quarterback if ours became damaged because none would have the wedge to pry open the gripper.

For this reason, an alternative design was implemented to release the ball from the Center’s gripper. This design is still entirely mechanical, except for the servo that drives the entire arm assembly. Once the arm lowers to a certain point, steel cables that are attached from the gripper jaws to a point near the bottom of the robot are put into tension and pull the jaws apart. The cables are highlighted in Figure 24.
The jaws are held together by tension springs that cause them to clamp onto the ball. Once the arm lowers to the point where the cables come into tension, the component of this force that is parallel and opposite the spring tension force causes the jaw to open as the servo drives the arm down. This tension point can be adjusted by tightening or loosening the turnbuckles at the base of the cables. Both the springs and the cables are highlighted in Figure 25.

This system makes it possible to carefully adjust the point of release so the ball is handed off smoothly to the Quarterback, but it is also able to drop the ball into the “basket” of a running back.

In order to cause the ball to be placed firmly into the Quarterback’s ball holder in the correct orientation for it to feed into the passing wheels, the alignment of the gripper to the Quarterback’s ball holder is crucial. It would be very difficult to build the parts of either robot to mate together perfectly straight from the shop, which is why several components have been made to be adjustable in order to come to a perfect fit between the two. Figure 26 illustrates the three translational degrees of freedom and one rotational degree of freedom present in the gripper on the center. The highlighted parts are the mechanisms that provide these degrees of freedom.
Another design component of the Center’s arm is that it is made to rise above the two foot height limit in place for the other robots so it does not get clipped during gameplay. It is held in place by driving the servo to the linkage’s toggle position, which is a natural resting point that does not require the servo to be constantly providing torque to the crank in order to hold the arm still as seen in Figure 27. This is very advantageous to extend the life of the servo.
1.4 Quarterback Baseplate and Chassis
The building process of the Quarterback took place at Jan Witte’s machine shop. It was a collaborative build between Colton Witte, Jeremy Nyikos, and Raihan Mir. The building started with the base plate, which was first machined using hand tools, but after further discussion it was determined that the base plate be remade using a much more precise milling machine. Figure 28 shows the hand cut base plate.

![Figure 28: Original quarterback base cut with hand tools](image)

The remade base plate, which is the one that was used for the Quarterback, was made with much more precision than the first one. The slots for the wheels were also taken out closer to the perimeter allowing for more space in the Center of the base plate where the battery was put. The slots for the side panels were not included in the remade base plate because the side panels were eliminated from the final Quarterback design. Figure 29 shows the final base plate for the Quarterback. Figure 29 also shows the four threaded rods (in the four corners of the base plate), which were used in place of the side panels. These threaded rods were used so the upper half of the Quarterback could be raised or lowered for alignment reasons.

![Figure 19: Final Quarterback baseplate](image)
Figure 30 shows the designed and built upper half of the Quarterback. The noticeable differences are that the turntable was eliminated and the launching mechanism was built in a fixed position. Also, the ball receptacle was changed from a gripper with wedges to a simple cupped holder. The position of the ball receptacle was also brought forward and the wedges, which make the receptacle level, were increased in size. The last noticeable difference is the passing wheels are fixed with the right set of wheels placed at a 45° angle.

Figure 30: Designed and built upper portion of Quarterback

Figure 31 shows the overall final design and the overall final build Quarterback. The big differences between the designed and built Quarterback can also be seen in Figure 31.

Figure 31: Final design and built Quarterback
2. DESCRIPTION OF SOFTWARE-PROGRAMMING PROCESS
The software used to program the electronic devices depends on the selected hardware. In the beginning, the choice was narrowed down to using a custom design microcontroller that was developed by Norte Dame, an Arduino Uno, and a Leaflab Maple. The programming environment (or integrated development environment – IDE) is what is used to program a microcontroller and generally a microcontroller manufacture will have a recommended IDE to use. The initial design was going to use an Arduino Uno because it is a widely recognized microcontroller and because all of the other electronic devices that were going to be used have Libraries designed for the Arduino microcontrollers. This means that the devices that connected to the Arduino are a ‘Plug & Play’ type device. First a device is wired together and then if desired a Library can be added to the Arduino’s standard Libraries. The Uno was advised against by Dr. Wang because of its 16 Mhz clock speed and 8 bit processing power. The team decided to use the Leaflab Maple microcontroller which has a 72 MHz clock speed and 32 bit processing power. The Maple microcontroller’s IDE is based off of the Arduino’s IDE and claimed to be just as user friendly; however this was not the case. The IDE itself was easy to install and use but trying to get the computer to recognize the microcontroller was difficult to say the least. The installation process that the Leaflab’s suggested for use was out of a forum. Once the device was finally able to be recognized by only one of the team’s laptops, the Maple was still temperamental when it came to downloading a program. Then a miracle came along, the Arduino Due was released and was in stock. The Due has a clock speed of 84 MHz clock speed and a 32 bit processor.

The Arduino Due comes with its own IDE which is what was used in this project. Since no one on the team had used this microcontroller or any of the other Arduino products, the examples and tutorials were used to learn the programming environment. After a few days of playing with LED’s, small motors, and a tiny servo it was time to develop a structure for each of the robots. The first step was to identify all of the variables, signal wires, the inputs and outputs (see the Table 1). Since the Quarterback has the most variables, signal wires, and I/O’s and should have all the same basic components as the other robots, it was decided to use its program as the model for the other robots. To aid in determining what was needed and to aid in the build process a schematic was drawn up as seen in Figure 32.
Figure 32: Quarterback Schematic

Table 1: Quarterback Variables

<table>
<thead>
<tr>
<th>Type</th>
<th>I/O or Internal Variable</th>
<th>Quarterback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commander</td>
<td>Input</td>
<td>Remote Controller</td>
</tr>
<tr>
<td>Sabertooth</td>
<td>Output</td>
<td>Front Motors</td>
</tr>
<tr>
<td>Sabertooth</td>
<td>Output</td>
<td>Rear Motors</td>
</tr>
<tr>
<td>Sabertooth</td>
<td>Output</td>
<td>Passing Motors</td>
</tr>
<tr>
<td>Sabertooth</td>
<td>Output</td>
<td>Ball Feed Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Front Right Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Front Left Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Rear Right Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Rear Left Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Right Passing Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Left Passing Motor</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Ball Feed</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Front Right Speed</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Front Left Speed</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Rear Right Speed</td>
</tr>
<tr>
<td>Int</td>
<td>Output</td>
<td>Rear Left Speed</td>
</tr>
<tr>
<td>Int</td>
<td>Input</td>
<td>Tackle Pin input</td>
</tr>
</tbody>
</table>
Compared to the Quarterback, the Center only has 2 wheels and 1 servo. The servo is used to raise and lower the arm to pass off the ball. The center’s servo had its difficulties that resulted from the library being designed for the UNO; however, the servo library was developed so that it was still useable. The library has a function that allows for any certain pulse width greater than 900 to be generated by the DUE. This allows for more control of the range the servo can travel and can also increase the resolution of the steps. Note that if the servo is moved while there is no power connected, the servo will no longer have the same 0° position. This is why the Center’s triggers are programmed to rotate the servo clockwise and counter clockwise. The servo cannot continuously rotate and it is limited from 0° to 400°. Once it reaches one of the extremes it must rotate in the opposite direction. See Figure 33 for a description of the Center’s remote controller.

![Center’s Remote Controller layout](image)

Figure 33: Center’s Remote Controller layout

The Xbee chips and the motor controllers are assigned a unique address which can be seen in Table 2. See Appendix for how to change addresses. Note that the Xbee chips have 2 chips per address so that they can communicate only with the other Xbee that has the same address.

<table>
<thead>
<tr>
<th>Device</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xbee (x high light)</td>
<td>1111</td>
</tr>
<tr>
<td>Xbee (b high light)</td>
<td>2222</td>
</tr>
<tr>
<td>Xbee (e high light)</td>
<td>3333</td>
</tr>
<tr>
<td>Sabertooth (Receiver)</td>
<td>128</td>
</tr>
<tr>
<td>Sabertooth (Center)</td>
<td>128</td>
</tr>
<tr>
<td>Sabertooth (Quarterback)</td>
<td>128, 129, 130, 131</td>
</tr>
</tbody>
</table>

Before the Sabertooth motor controllers were assigned an address, the team explored some of the other methods for controlling the Sabertooth such as Mode 1- Analog Input, Mode 2-R/C Input, Mode 3-
Simplified Serial, and Mode 4-Packetized Serial. Initially Mode 2 seemed to be the best route because each motor would have its own signal wire that would have a PWM signal. However, since the motor controller was designed to receive a 0V to 5V signal there was going to be problems resulting from the microcontroller since the I/O’s on the Due operate on a 0V to 3.3V system. This would have required the use of a logic level shifter to increase the voltage to the desired voltage. The next option was to use Mode 3-Simplified Serial as it appeared to be a viable option for both the Center and Receiver since each of them would only need one motor controller. The Quarterback however needed to have 4 motor controllers and the Due only has 4 predefined UART’s. This posed a problem since a fifth UART would be needed for the Xbee chip. After some testing were completed using the Simplified Serial, the team began testing the Packetized Serial option. This only required a single transmission line which meant fewer wires and easier debugging as well as simplified programming.

At this time, the Commander remote controller was being tested using tutorials found in the Commander Library. Note that the initial testing was done using Arduino UNO. The testing did not take long as the remote controller was successfully able to turn on and off different LED on a breadboard depending upon what button was pressed. Next, the joysticks were used to vary the brightness of an LED.

Next, the team tested how to simply turn on and off a motor using the remote controller, motor controller, and the DUE. This is when some crucial problems were brought to light about the libraries that were imported for the remote controller and the motor controller. The key problem was that the libraries were designed to work with the UNO, which only has 1 UART. So both of the libraries would only work on Serial Port 0 and both devices required different Baud Rates as well. After identifying the problem, the solution was a relatively easy fix. In Commander.cpp file, wiring.h was changed to Arduino.h and Serial0 to Serial1 or any other desired Serial port. For testing, first one motor was hooked up and then 2 motors were hooked up. Below is an overall flow of the drive train system.

1. Check if there is a new message on the UART
2. Read the data from the UART
   a. The library handles how the message is broken up, for further details see the Commander’s data sheet
3. Call the motor controller operator and give it the remote controller data and to which motor.

The next task was to develop an equation of motion for the quarterback’s omni-directional wheels. The team found several papers about using position, speed, and various other systems to monitor the path traveled using omni-directional wheels \(^6\). However, these papers were truly unnecessary for this implementation of omni-direction wheels since a human operator who can compensate for any slippage of the wheels is used. See Table 3 below for the original equations and the final product.
Table 3: Testing for the QB’s drivetrain

<table>
<thead>
<tr>
<th></th>
<th>Old</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Front_Right_Speed = round( .707*( WalkH + WalkV) + LookH));</td>
<td>Front_Right_Speed = round((boost/100)<em>(( WalkH + WalkV) + .75</em>LookH));</td>
</tr>
<tr>
<td></td>
<td>Front_Left_Speed = round( 707*(-WalkH + WalkV) - LookH));</td>
<td>Front_Left_Speed = round((boost/100)<em>((-WalkH + WalkV) - .75</em>LookH));</td>
</tr>
<tr>
<td></td>
<td>Rear_Right_Speed = round( .707*( WalkH + WalkV) - LookH));</td>
<td>Rear_Right_Speed = round((boost/100)<em>(( WalkH + WalkV) - .75</em>LookH));</td>
</tr>
<tr>
<td></td>
<td>Rear_Left_Speed = round( .707*(- WalkH + WalkV) + LookH));</td>
<td>Rear_Left_Speed = round((boost/100)<em>((-WalkH + WalkV) + .75</em>LookH));</td>
</tr>
</tbody>
</table>

A layout of the Quarterback’s handheld controller is shown in Figure 34. Originally the .707 factor was thought to account for the wheels being set at a 45° angle. After playing with the robot it was determined that it wasn’t a necessary scalar. In an intermediate step to the final equation, the .707 factor was reduced to .3 to help increase the handling and control ability. By doing this, the robot was slow and could easily be run down by a defender. For this reason, an added feature was implemented that allowed the speed to be increased by the user. This was done by pushing forward on the right joystick (look joystick); when the value sent back from lookV of the joystick exceeds 75 it then uses that value divided by 128 as the new boost value but when it is below a value of 75 the boost defaults to .25.

![Figure 34: Quarterback trigger functions and drive train function](image)

The buttons just above the right joystick are programmed to control the speed of the passing wheel. The only issue was finding a speed for the motors that achieved a 5ft, 10ft, and 20ft pass. The left most button above the left joystick is programmed to stop the passing wheels. The 2 triggers on top of the remote controller are used to advance and retract the ball retention mechanism into and away from the passing wheels. See Figure 35 for Quarterback button layout.
Another major piece of hardware is the accelerometer kit that was provided by Norte Dame. Interfacing the accelerometer was quite easy. The only difficulty came from Norte Dame saying that we needed to upload a new set of binary’s to it. It was soon discovered that a special programming unit was required to be able to program the accelerometer, and when questions were raised, that was when someone at Norte Dame realized that these new binaries had already been upload. This was when the team also learned what signal should be appearing on the output pins. The output should be a high signal except when it is in a tackled state at which time the accelerometer outputs a low signal. The high signal that is expected form the accelerometer is 3.3V and the low signal should be below .3V.

After the game was over with and there was a little free time, a much better program was written for the receiver handle controller. Instead of driving the receiver like a tank (left joystick controls left wheel, right joystick controls right wheel), the left joystick would control how fast the receiver would drive forward or backward. See Figure 36 below.
The left joystick would reduce the speed of the wheel based upon the left and right movement of the joystick. When the joystick is pushed to the left, the left wheel speed would be reduced and if the joystick is pushed to the right, the right wheels speed would be reduced. This allowed for much better control at high speeds. However, the maneuverability at slow speeds decreased as well. It was agreed that in game time situations speed and ability to maintain control at high speed was more favorable.
Section III
1. TESTING
Testing of the robots is broken down into three main sections. These sections include locomotion, ball transfer and wireless communication.

1.1 Parameters
Here is a list of parameters provided within the design section.

1.1.1 Locomotion
For locomotion, only the receiver and center need to be tested. The speed of the quarterback does not need full testing due to it having a focus on maneuverability.

- Center
  - Travel 50 feet within 8 seconds starting from rest.
- Receiver
  - Travel 50 feet within 5 seconds starting from rest.

1.1.2 Ball Transfer
Since ball transfer is the foundation of robotic football, it is important that this section gets tested efficiently.

- Center to Quarterback
  - Center should deliver ball within 20 seconds to the quarterback with a success rate of 75%.
- Quarterback to Receiver
  - Able to complete a pass 65% of the time.

1.1.3 Wireless Communication
It is important to ensure that the Xbees will be able to retain constant communication from anywhere on the field. A pair of Xbee Communication Chips must be able to pair anywhere from within the robotic football field range (94 feet long and 50 feet wide).

- Xbee Communication Chip
  - A pair of Xbee Communication Chips should retain constant communication from corner to corner of the robotic football playing field (106.47 feet).
1.2 Testing Procedure

1.2.1 Locomotion
To test the locomotion of a robot, two cones are placed to establish a start from rest point and a finish point 50 feet apart. A stopwatch is used to time the robot being tested. The turf on which the robots travel on will vary speed. The tests are performed on a smooth surface which allows for less gripping than the actual playing field, resulting in slower results.

- **Center**
  - The center is able to start from rest and travel 50 feet within 6.7 seconds.
- **Receiver**
  - The receiver is able to start from rest and travel 50 feet within 4.7 seconds.

1.2.2 Ball Transfer
For the ball transfer between the center and quarterback, the robots are set five feet apart. The time recorded is from when the robots start to move to after they make contact and the ball is snapped. For the ball transfer between the quarterback and receiver, the robots are set ten feet apart while the operator makes a pass with a set speed.

- **Center to Quarterback**
  - On average, the ball is snapped in 5 seconds.
- **Quarterback to Receiver**
  - On average, the pass is completed 4 out of 5 trials.

1.2.3 Wireless communication
To test for the wireless communication between an Xbee pair, the center is taken out to a parking lot to see how far of a distance it can travel from the operator while retaining constant communication.

- **Xbee Communication Chip**
  - The Xbee chip pair is able to retain constant communication within 110 feet.
Section IV
1. EVALUATION AND RECOMMENDATIONS

All parameters from the design phase have been met. The robots were taken to the game to compete. There were no malfunctions and the robots were out on the field for every play. No repairs were necessary in the game. The batteries lasted the whole game without needing swapped.

1.1 Receiver Recommendations

It is recommended to change the drivetrain of the receiver by (a) moving the two wheels in the rear and using only a single caster in the front of the robot or (b) using omni-wheels for more maneuverability. The netting system can be improved with a better design. Even though the cross-netting system design is functional, it is not easy to implement.

1.2 Center Recommendations

It is recommended to change the eyebolts used for the strain cables to pulleys. Due to tension, the steel cable started fraying and was already replaced once. Also, the Center is the only player that does not use a prototype board for the Xbee and microcontroller connections. It is suggested to replace the existing breadboard with a prototype board for more secure wiring. In fact, if the Center is tackled too hard the wire connections may come loose. The team originally used a prototype board for the Center; however, while troubleshooting it was found that the prototype board was not working correctly and decided to replace it with a breadboard due to time constrain.

1.3 Quarterback Recommendations

It is recommended to improve the design of the omni-wheels to eliminate any vibration issues. Also, it is suggested to add a turntable to the Quarterback’s top plate to facilitate aiming capabilities and design an aiming system such as laser to determine its orientation. The current ball holder is made of a plastic cup toughened by electrical tape; this design can be improved to a more sophisticated one. The team did not have enough time to pursue a tracking system. In the future, it is advised to include a tracking and positioning system in the design of the Quarterback. This will greatly benefit during game time to complete passes.

1.4 Additional Recommendations

The following is a list of general recommendations that will help in simplifying the manufacturing process and will guarantee a successful design:

- Set up connections with local Machine Shops to gain access to tools
- Gain connections or access in a timely manner
- Design simple, yet functional parts
- Design for Aesthetics
- Move up timeline of milestones to secure more machining, debugging, and practice time
• Look into using Xbox Controller with modified Xbee
• Take plenty of spare parts and tools to competition
• PRACTICE
Conclusions
We successfully designed and implemented three complete football playing robots that competed at the intercollegiate mechatronic game at Notre Dame University on April 19th, 2013. Throughout the building process, we made changes to the original system design to overcome unforeseen issues. However, we were able to solve these issues in order to meet all set requirements and improve the proficiency of each robot. Finally, we were able to stay within our budget.
References
Appendix
**Arduino 1.5.2 Software Installation**

Download Arduino software, version 1.5.2, from Arduino website. This is the only version that supports Arduino Due board. Additional driver installation may be required depending on operating system.

For Windows OS

- Download Windows version of Arduino 1.5.2 and unzip the files
- Connect the Due to your computer with a USB cable via the programming port
- Windows will initiate driver installation process, but it won’t know its location
- Open the Device Manager
- Look for an open port named *Arduino Due Prog. Port*
- Right click on the *Arduino Due Prog. Port* and choose *Update Driver Software*
- Select the *Browse my computer for Driver software* option
- Navigate to the folder with the Arduino IDE. Locate and select the *Drivers* folder in the main Arduino folder. Press *OK* and *Next* to proceed
- If you are prompted with a warning dialog about not passing Windows Logo testing, click *Continue Anyway*
- Windows will take over the installation

**Adding Sabertooth and Commander Libraries to Arduino IDE**


- Un-compress the Zip file containing the library
- Drag the Library folder in your Arduino Library folder. Under Windows, it will likely be called “My Documents\Arduino\libraries”

**Software Modifications – Commander Library**

In the .cpp file of Commander Library, change wiring.h to Arduino.h.

In the .cpp file of Commander Library, change serial number to 1. The handheld controllers communicate with Arduino through port TX1 and RX1. The motor controllers use serial port TX0 and RX0. If wanting to use a different port, make adequate changes to Commander.cpp file.

**Uploading Code to the Due**

Follow these directions to upload code to the Due board

- Attach the USB micro side of the USB cable to the Due’s Programming port (port closer to the DC power connector)
- Open the Arduino IDE
- Choose **Arduino Due (Programming port)** from the **Tools > Board** menu in the Arduino IDE, and select the correct serial port from the **Tools > Serial Port** menu.
- Click on the check-sign button to verify your code
- Click on the right-arrow button to upload your code

**X-CTU Software Guide**

This software is used to program the Xbee modules

- Download X-CTU Software
- Connect the Xbee (attached to its shield) to your computer with a USB cable
- Select the USB Port the Xbee is connected to under **PC Settings > Select Com Port**
- Select the Baud the Xbee is programmed for under **PC Settings > Select Com Port > Baud**. Note: if the Xbee module is new, the standard baud rate is 9600. The current modules are programmed at 38400.
- Click on **Test/Query** to check communication is available
- To program the Xbee module, click on **Modem Configuration** Tab
- Click on **Read** under **Modem Parameter and Firmware**
- Click on **Baud Rate** under **Serial Interfacing** to select your desired baud rate from the drop-down menu
- To pair 2 Xbee, assign the same numerical address to both. Click **ID-Modem VID** under **Networking/Security** and type a 4-digit address-ID
- When finishing the module configuration, click on **Write** to save the settings
Sabertooth 2x25 Address Byte Configuration
Address bytes are set by switches 4, 5, and 6. Addresses start at 128 and go to 135. The switch settings for the addresses are shown in the chart below.
<table>
<thead>
<tr>
<th>Part</th>
<th>Supplier</th>
<th>Part No.</th>
<th>Qty</th>
<th>Cost(unit) [$]</th>
<th>Cost(total) [$]</th>
<th>Total Cost [$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wheel, 4-7/8, key Mount</td>
<td>BaneBots</td>
<td>T80P-494BO-KS4</td>
<td>4</td>
<td>6.80</td>
<td>27.20</td>
<td>776.86</td>
</tr>
<tr>
<td>RSS40 Motors</td>
<td>BaneBots</td>
<td>M5-RSS40-12</td>
<td>2</td>
<td>6.25</td>
<td>12.50</td>
<td></td>
</tr>
<tr>
<td>P60 Gearbox, 16:1</td>
<td>BaneBots</td>
<td>P60K-44-0004</td>
<td>2</td>
<td>49.05</td>
<td>98.10</td>
<td></td>
</tr>
<tr>
<td>Machine Key (1/8 x 1/8 x 5/8)</td>
<td>BaneBots</td>
<td>MKS-12120625</td>
<td>2</td>
<td>0.50</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>10-32 0.75&quot; Screws</td>
<td>Home Depot</td>
<td>N/A</td>
<td>8</td>
<td>0.25</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>#10 Washers</td>
<td>Home Depot</td>
<td>N/A</td>
<td>8</td>
<td>0.05</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td>6-32 1.00&quot; Screws and Nuts</td>
<td>Lowe's</td>
<td>N/A</td>
<td>8</td>
<td>0.07</td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td>8-32 1.00&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>36</td>
<td>0.16</td>
<td>5.76</td>
<td></td>
</tr>
<tr>
<td>8-32 2.00&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>15</td>
<td>0.25</td>
<td>3.75</td>
<td></td>
</tr>
<tr>
<td>8-32 3.00&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>8</td>
<td>0.33</td>
<td>2.64</td>
<td></td>
</tr>
<tr>
<td>Industrial Strength Velcro; 2x4 in</td>
<td>Home Depot</td>
<td>N/A</td>
<td>2</td>
<td>5.94</td>
<td>11.88</td>
<td></td>
</tr>
<tr>
<td>#8 Washers</td>
<td>Home Depot</td>
<td>N/A</td>
<td>44</td>
<td>0.05</td>
<td>2.20</td>
<td></td>
</tr>
<tr>
<td>0.5&quot; HDPE</td>
<td>McMaster-Carr</td>
<td>8619K477</td>
<td>300 sq.in</td>
<td>115.48</td>
<td>15.10</td>
<td></td>
</tr>
<tr>
<td>0.25&quot; HDPE</td>
<td>McMaster-Carr</td>
<td>8619K166</td>
<td>440 sq.in</td>
<td>107.47</td>
<td>10.26</td>
<td></td>
</tr>
<tr>
<td>70 durometer black epdm rubber bumper</td>
<td>Wefco Rubber</td>
<td>4134</td>
<td>6 ft</td>
<td>2.00</td>
<td>12.00</td>
<td></td>
</tr>
<tr>
<td>Illuminated Toggle Switch</td>
<td>RobotShop</td>
<td>RB-Spa-709</td>
<td>1</td>
<td>2.95</td>
<td>2.95</td>
<td></td>
</tr>
<tr>
<td>1.00&quot; dia. x 1.00' PVC Pipe (LED Diffuser)</td>
<td>Home Depot</td>
<td>N/A</td>
<td>1</td>
<td>2.99</td>
<td>2.99</td>
<td></td>
</tr>
<tr>
<td>Flange Mount Ball Transfer</td>
<td>McMaster-Carr</td>
<td>T80P-494BO-KS3</td>
<td>2</td>
<td>6.80</td>
<td>13.60</td>
<td></td>
</tr>
<tr>
<td>Polypropylene Draw Latch</td>
<td>McMaster-Carr</td>
<td>1891A48</td>
<td>4</td>
<td>10.98</td>
<td>10.98</td>
<td></td>
</tr>
<tr>
<td>Cable Ties</td>
<td>Home Depot</td>
<td>N/A</td>
<td>50</td>
<td>0.10</td>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>Flange Mount Ball Transfer</td>
<td>McMaster-Carr</td>
<td>T80P-494BO-KS3</td>
<td>2</td>
<td>6.80</td>
<td>13.60</td>
<td></td>
</tr>
<tr>
<td>1/2 in. x 10 ft. PVC</td>
<td>Home Depot</td>
<td>530048</td>
<td>1</td>
<td>1.47</td>
<td>1.47</td>
<td></td>
</tr>
<tr>
<td>Status LED</td>
<td>University of Notre Dame</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Accelerometer</td>
<td>University of Notre Dame</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>L-Brackets</td>
<td>Colton Witte</td>
<td>N/A</td>
<td>8</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Arduino DUE</td>
<td>Arduino</td>
<td>RB-Ard-40</td>
<td>1</td>
<td>49.95</td>
<td>49.95</td>
<td></td>
</tr>
<tr>
<td>Arduino Mege Prototype Shield</td>
<td>DFRobot</td>
<td>DFR0016</td>
<td>1</td>
<td>6.80</td>
<td>6.80</td>
<td></td>
</tr>
<tr>
<td>XBee Module to DIP Adapter</td>
<td>NKC Electronics</td>
<td>SMD-0101</td>
<td>1</td>
<td>3.45</td>
<td>3.45</td>
<td></td>
</tr>
<tr>
<td>Sabertooth 2x25</td>
<td>Dimension Engineering</td>
<td>RB-Dim-23</td>
<td>1</td>
<td>124.99</td>
<td>124.99</td>
<td></td>
</tr>
<tr>
<td>ArbotiX Commander v2.0</td>
<td>ArbotiX</td>
<td>ASM-ARBX-CMD-V2</td>
<td>1</td>
<td>59.95</td>
<td>59.95</td>
<td></td>
</tr>
<tr>
<td>NiMH Battery; 12 V, 10000 mAh (10 X D)</td>
<td>Mega Batteries</td>
<td>13715</td>
<td>1</td>
<td>179.99</td>
<td>179.99</td>
<td></td>
</tr>
<tr>
<td>Energizer AA Batteries</td>
<td>Walmart</td>
<td>N/A</td>
<td>1</td>
<td>7.54</td>
<td>7.54</td>
<td></td>
</tr>
<tr>
<td>NiMH Battery; 7.2 V, 4500 mAh</td>
<td>Mega Batteries</td>
<td>14673</td>
<td>1</td>
<td>45.95</td>
<td>45.95</td>
<td></td>
</tr>
<tr>
<td>XBee 1mW Communication Module</td>
<td>Digi</td>
<td>PX-32405</td>
<td>2</td>
<td>21.95</td>
<td>43.90</td>
<td></td>
</tr>
<tr>
<td>Part</td>
<td>Supplier</td>
<td>Part No.</td>
<td>Qty</td>
<td>Cost(Unit) [$]</td>
<td>Cost(total) [$]</td>
<td>Total Cost [$]</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>-----</td>
<td>----------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>0.5” HDPE</td>
<td>McMaster-Carr</td>
<td>8619K477</td>
<td>460 sq.in</td>
<td>115.48</td>
<td>23.06</td>
<td>1130.727</td>
</tr>
<tr>
<td>0.25” HDPE</td>
<td>McMaster-Carr</td>
<td>8619k166</td>
<td>400 sq.in</td>
<td>107.47</td>
<td>9.33</td>
<td></td>
</tr>
<tr>
<td>RS540 Motor</td>
<td>BaneBots</td>
<td>M5-RS540-12</td>
<td>2</td>
<td>6.25</td>
<td>12.5</td>
<td></td>
</tr>
<tr>
<td>P60 Gearbox, 16:1</td>
<td>Banebots</td>
<td>P60K-44-0004</td>
<td>2</td>
<td>49.05</td>
<td>98.1</td>
<td></td>
</tr>
<tr>
<td>Wheel; 4-7/8, key mount</td>
<td>Banebots</td>
<td>T80P-494BO-K54</td>
<td>4</td>
<td>6.8</td>
<td>27.2</td>
<td></td>
</tr>
<tr>
<td>Flange Mount Ball Transfer</td>
<td>McMaster-Carr</td>
<td>T80P-494BO-K53</td>
<td>2</td>
<td>6.8</td>
<td>13.6</td>
<td></td>
</tr>
<tr>
<td>Fully Threaded Rod; 24 in, 3/8-24</td>
<td>McMaster-Carr</td>
<td>90032A031</td>
<td>4</td>
<td>2.63</td>
<td>10.52</td>
<td></td>
</tr>
<tr>
<td>Nuts; 3/8-24</td>
<td>Lowe's</td>
<td>N/A</td>
<td>24</td>
<td>0.56</td>
<td>13.44</td>
<td></td>
</tr>
<tr>
<td>2 in Narrow Hinge</td>
<td>Lowe's</td>
<td>N/A</td>
<td>4</td>
<td>1.19</td>
<td>4.76</td>
<td></td>
</tr>
<tr>
<td>Clevis Pin; 1/2 D; 4-1/4 L</td>
<td>McMaster-Carr</td>
<td>92735A940</td>
<td>1</td>
<td>10.63</td>
<td>10.63</td>
<td></td>
</tr>
<tr>
<td>Acetal Spacer; 1/2 ID; 1/2 L</td>
<td>McMaster-Carr</td>
<td>93717A636</td>
<td>1</td>
<td>4.79</td>
<td>4.79</td>
<td></td>
</tr>
<tr>
<td>Acetal Spacer; 1/2 ID; 3/8 L</td>
<td>McMaster-Carr</td>
<td>93717A641</td>
<td>2</td>
<td>5.22</td>
<td>10.44</td>
<td></td>
</tr>
<tr>
<td>Clevis Pin; 1/4 D; 1-1/2 L</td>
<td>McMaster-Carr</td>
<td>93890A580</td>
<td>1</td>
<td>4.34</td>
<td>4.34</td>
<td></td>
</tr>
<tr>
<td>Acetal Spacer; 1/4 ID; 1/4 L</td>
<td>McMaster-Carr</td>
<td>93890A416</td>
<td>1</td>
<td>3.06</td>
<td>3.06</td>
<td></td>
</tr>
<tr>
<td>Clevis Pin; 1/4 D; 1-1/4 L</td>
<td>McMaster-Carr</td>
<td>93890A576</td>
<td>1</td>
<td>4.32</td>
<td>4.32</td>
<td></td>
</tr>
<tr>
<td>Fully Threaded Rod; 7 in, 10-32</td>
<td>McMaster-Carr</td>
<td>91565A868</td>
<td>2</td>
<td>5.75</td>
<td>11.5</td>
<td></td>
</tr>
<tr>
<td>Fully Threaded Rod; 3 1/2 (cut to length), 10-32</td>
<td>McMaster-Carr</td>
<td>91565A868</td>
<td>2</td>
<td>5.75</td>
<td>11.5</td>
<td></td>
</tr>
<tr>
<td>Nuts; 10-32</td>
<td>Lowe's</td>
<td>N/A</td>
<td>18</td>
<td>0.05</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>Shoulder Screw; 10-32, 1/4 D; 3/8 L</td>
<td>McMaster-Carr</td>
<td>93996A842</td>
<td>8</td>
<td>5.9</td>
<td>47.2</td>
<td></td>
</tr>
<tr>
<td>Shoulder Screw; 10-32, 1/4 D, 1/8 L</td>
<td>McMaster-Carr</td>
<td>93996A831</td>
<td>4</td>
<td>5.31</td>
<td>21.24</td>
<td></td>
</tr>
<tr>
<td>Shoulder Screw; 10-32, 1/4 D, 1/4 L</td>
<td>McMaster-Carr</td>
<td>93996A836</td>
<td>2</td>
<td>5.61</td>
<td>11.22</td>
<td></td>
</tr>
<tr>
<td>Brass Press-Fit Expansion Insert for Plastics; 10-32</td>
<td>McMaster-Carr</td>
<td>N/A</td>
<td>16</td>
<td>8.5</td>
<td>8.5</td>
<td></td>
</tr>
<tr>
<td>Eye Bolt; 5/32 X 1 5/8</td>
<td>Lowe’s</td>
<td>N/A</td>
<td>4</td>
<td>0.49</td>
<td>1.96</td>
<td></td>
</tr>
<tr>
<td>Large Eye Screw</td>
<td>Lowe’s</td>
<td>N/A</td>
<td>2</td>
<td>0.42</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>Small Eye Screw</td>
<td>Lowe’s</td>
<td>N/A</td>
<td>3</td>
<td>0.28</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>Aluminum Tube</td>
<td>Lowe’s</td>
<td>N/A</td>
<td>1</td>
<td>2.19</td>
<td>2.19</td>
<td></td>
</tr>
<tr>
<td>Threaded Steel Cable</td>
<td>Home Depot</td>
<td>N/A</td>
<td>2</td>
<td>0.85</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>Aluminum Ferrule</td>
<td>Home Depot</td>
<td>N/A</td>
<td>4</td>
<td>0.82</td>
<td>3.28</td>
<td></td>
</tr>
<tr>
<td>Turnbuckle; 3/16 X 5 1/2</td>
<td>Lowe’s</td>
<td>N/A</td>
<td>2</td>
<td>4.94</td>
<td>9.88</td>
<td></td>
</tr>
<tr>
<td>Quick Link</td>
<td>Home Depot</td>
<td>N/A</td>
<td>2</td>
<td>1.64</td>
<td>3.28</td>
<td></td>
</tr>
<tr>
<td>Polypropelene Draw Latch</td>
<td>McMaster-Carr</td>
<td>1891A48</td>
<td>4</td>
<td>10.98</td>
<td>43.92</td>
<td></td>
</tr>
<tr>
<td>Tension Spring</td>
<td>Lowe’s</td>
<td>N/A</td>
<td>2</td>
<td>1.64</td>
<td>3.28</td>
<td></td>
</tr>
<tr>
<td>Industrial Strength Velcro; 2x4 in</td>
<td>Home Depot</td>
<td>N/A</td>
<td>2</td>
<td>5.94</td>
<td>11.88</td>
<td></td>
</tr>
<tr>
<td>Kwik Clip; 3/8 in</td>
<td>Home Depot</td>
<td>N/A</td>
<td>3</td>
<td>0.54</td>
<td>1.62</td>
<td></td>
</tr>
<tr>
<td>Part</td>
<td>Supplier</td>
<td>Part No.</td>
<td>Qty.</td>
<td>Cost(Unit) [$]</td>
<td>Cost(total) [$]</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------</td>
<td>----------</td>
<td>------</td>
<td>-----------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>Cable Ties</td>
<td>Home Depot</td>
<td>N/A</td>
<td>10</td>
<td>0.0137</td>
<td>0.137</td>
<td></td>
</tr>
<tr>
<td>Bracket, 90º</td>
<td>Lowe's</td>
<td>N/A</td>
<td>3</td>
<td>2.95</td>
<td>8.85</td>
<td></td>
</tr>
<tr>
<td>Illuminated Toggle Switch</td>
<td>RobotShop</td>
<td>RB-Spa-709</td>
<td>1</td>
<td>2.95</td>
<td>2.95</td>
<td></td>
</tr>
<tr>
<td>8-32 1.00&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>28</td>
<td>0.16</td>
<td>4.48</td>
<td></td>
</tr>
<tr>
<td>1.00&quot; dia. x 1.00' PVC Pipe (LED Diffuser)</td>
<td>Home Depot</td>
<td>N/A</td>
<td>1</td>
<td>2.99</td>
<td>2.99</td>
<td></td>
</tr>
<tr>
<td>70 durometer black epdm rubber bumper</td>
<td>Wefco Rubber</td>
<td>4134</td>
<td>6 ft</td>
<td>2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Status LED</td>
<td>University of Notre Dame</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Accelerometer</td>
<td>University of Notre Dame</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Arduino DUE</td>
<td>Arduino</td>
<td>RB-Ard-40</td>
<td>1</td>
<td>49.95</td>
<td>49.95</td>
<td></td>
</tr>
<tr>
<td>Arduino Mega Prototype Shield</td>
<td>DFRobot</td>
<td>DFR0016</td>
<td>1</td>
<td>6.8</td>
<td>6.8</td>
<td></td>
</tr>
<tr>
<td>XBee Module to DIP Adapter</td>
<td>NKC Electronics</td>
<td>SMD-0101</td>
<td>1</td>
<td>3.45</td>
<td>3.45</td>
<td></td>
</tr>
<tr>
<td>Sabertooth 2x25</td>
<td>Dimension Engineering</td>
<td>RB-Dim-23</td>
<td>1</td>
<td>124.99</td>
<td>124.99</td>
<td></td>
</tr>
<tr>
<td>ArbotiX Commander v2.0</td>
<td>ArbotiX</td>
<td>ASM-ARBX-CMD-V2</td>
<td>1</td>
<td>59.95</td>
<td>59.95</td>
<td></td>
</tr>
<tr>
<td>NiMH Battery; 12 V, 10000 mAh (10 X D)</td>
<td>Mega Batteries</td>
<td>13715</td>
<td>1</td>
<td>179.99</td>
<td>179.99</td>
<td></td>
</tr>
<tr>
<td>NiMH Battery; 7.2 V, 4500 mAh</td>
<td>Mega Batteries</td>
<td>14673</td>
<td>1</td>
<td>45.95</td>
<td>45.95</td>
<td></td>
</tr>
<tr>
<td>Energizer AA Batteris</td>
<td>Walmart</td>
<td>N/A</td>
<td>1</td>
<td>7.54</td>
<td>7.54</td>
<td></td>
</tr>
<tr>
<td>XBee 1mW Communication Module</td>
<td>Digi</td>
<td>PX-32405</td>
<td>2</td>
<td>21.95</td>
<td>43.9</td>
<td></td>
</tr>
<tr>
<td>Top mount Servo</td>
<td>ServoCity</td>
<td>SPGS685A-360</td>
<td>1</td>
<td>129.98</td>
<td>129.98</td>
<td></td>
</tr>
</tbody>
</table>
### Quarterback BOM

<table>
<thead>
<tr>
<th>Part</th>
<th>Supplier</th>
<th>Part No.</th>
<th>Qty.</th>
<th>Cost(unit) [$]</th>
<th>Cost(total) [$]</th>
<th>Total [$]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4&quot; Plastic Omni Wheel</td>
<td>AndyMark</td>
<td>am-0383</td>
<td>4</td>
<td>19.00</td>
<td>76</td>
<td>1330.13</td>
</tr>
<tr>
<td>500 Key Hub</td>
<td>AndyMark</td>
<td>am-0077</td>
<td>4</td>
<td>15.00</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>RS540 Motors</td>
<td>BaneBots</td>
<td>MS-RS540-12</td>
<td>4</td>
<td>6.25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>P60 Gearbox, 16:1</td>
<td>BaneBots</td>
<td>P60K-44-0004</td>
<td>4</td>
<td>49.05</td>
<td>196.2</td>
<td></td>
</tr>
<tr>
<td>Machine Key (1/8 x 1/8 x 5/8)</td>
<td>BaneBots</td>
<td>MKSS-12120625</td>
<td>4</td>
<td>0.50</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10-32 0.75&quot; Screws</td>
<td>Home Depot</td>
<td>N/A</td>
<td>16</td>
<td>0.25</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>#10 Washers</td>
<td>Home Depot</td>
<td>N/A</td>
<td>16</td>
<td>0.05</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>6-32 1.00&quot; Screws and Nuts</td>
<td>Lowe's</td>
<td>N/A</td>
<td>24</td>
<td>0.07</td>
<td>1.68</td>
<td></td>
</tr>
<tr>
<td>10-32 0.5&quot; Screws</td>
<td>Home Depot</td>
<td>N/A</td>
<td>4</td>
<td>0.25</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>#10 Washers</td>
<td>Menards</td>
<td>N/A</td>
<td>8</td>
<td>0.03</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>0.5&quot; Washers</td>
<td>Jan Witte</td>
<td>N/A</td>
<td>16</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>10-32 0.75&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>10</td>
<td>0.25</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>4&quot; L-Bracket</td>
<td>Jan Witte</td>
<td>N/A</td>
<td>4</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>6.5&quot; L-Bracket</td>
<td>Jan Witte</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>5/16-18 LockWashers</td>
<td>Menards</td>
<td>N/A</td>
<td>12</td>
<td>0.08</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>5/16-18 LockNuts</td>
<td>Menards</td>
<td>N/A</td>
<td>12</td>
<td>0.11</td>
<td>1.32</td>
<td></td>
</tr>
<tr>
<td>5/16-18 WingNuts</td>
<td>Menards</td>
<td>N/A</td>
<td>4</td>
<td>0.11</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>5/16-18 8&quot; Threaded Rod</td>
<td>Menards</td>
<td>N/A</td>
<td>4</td>
<td>0.44</td>
<td>1.76</td>
<td></td>
</tr>
<tr>
<td>Illuminated Toggle Switch</td>
<td>RobotShop</td>
<td>RB-Spa-709</td>
<td>1</td>
<td>2.95</td>
<td>2.95</td>
<td></td>
</tr>
<tr>
<td>1.00&quot; dia. x 1.00&quot; PVC Pipe (LED Diffuser)</td>
<td>Home Depot</td>
<td>N/A</td>
<td>1</td>
<td>2.99</td>
<td>2.99</td>
<td></td>
</tr>
<tr>
<td>10-32 0.5&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>16</td>
<td>0.25</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1.25&quot; x 1.25&quot; L-Bracket</td>
<td>Lowe's</td>
<td>N/A</td>
<td>8</td>
<td>0.62</td>
<td>4.96</td>
<td></td>
</tr>
<tr>
<td>10-32 0.75&quot; Screws</td>
<td>Home Depot</td>
<td>N/A</td>
<td>16</td>
<td>0.25</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>RS540 Motors</td>
<td>BaneBots</td>
<td>MS-RS540-12</td>
<td>2</td>
<td>6.25</td>
<td>12.5</td>
<td></td>
</tr>
<tr>
<td>P60 Gearbox, 4:1</td>
<td>BaneBots</td>
<td>P60K-4-0004</td>
<td>2</td>
<td>44.5</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>Wheel, 4-7/8, key Mount</td>
<td>BaneBots</td>
<td>T80P-49480-K54</td>
<td>4</td>
<td>6.8</td>
<td>27.2</td>
<td></td>
</tr>
<tr>
<td>10-32 0.50&quot; Screws</td>
<td>Home Depot</td>
<td>N/A</td>
<td>2</td>
<td>0.16</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>#10 Washers</td>
<td>Menards</td>
<td>N/A</td>
<td>4</td>
<td>0.08</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td>10-32 1.00&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>6</td>
<td>0.16</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>10-32 1.25&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>4</td>
<td>0.25</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>#10 Washers</td>
<td>Menards</td>
<td>N/A</td>
<td>40</td>
<td>0.08</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>8.00&quot; L-Bracket</td>
<td>Jan Witte</td>
<td>N/A</td>
<td>2</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>1.50&quot; x 0.75&quot; L-Bracket</td>
<td>Colton Witte</td>
<td>N/A</td>
<td>8</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>0.75&quot; x 0.75&quot; L-Bracket</td>
<td>Lowe's</td>
<td>N/A</td>
<td>4</td>
<td>0.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8-32 1.00&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>24</td>
<td>0.16</td>
<td>3.84</td>
<td></td>
</tr>
<tr>
<td>Part</td>
<td>Supplier</td>
<td>Part No.</td>
<td>Qty.</td>
<td>Cost(unit) [$]</td>
<td>Cost(total) [$]</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
<td>--------------</td>
<td>------</td>
<td>----------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>8-32 0.5&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>8</td>
<td>0.16</td>
<td>1.28</td>
<td></td>
</tr>
<tr>
<td>#8 Washers</td>
<td>Home Depot</td>
<td>N/A</td>
<td>12</td>
<td>0.08</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>RS540 Motors</td>
<td>BaneBots</td>
<td>MS-RS540-12</td>
<td>1</td>
<td>6.25</td>
<td>6.25</td>
<td></td>
</tr>
<tr>
<td>10-32 2.50&quot; Screw</td>
<td>Home Depot</td>
<td>N/A</td>
<td>2</td>
<td>0.49</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>Steel 12&quot; long 3/8&quot;-8 lead screw</td>
<td>McMaster-Carr</td>
<td>98935A706</td>
<td>1</td>
<td>8.86</td>
<td>8.86</td>
<td></td>
</tr>
<tr>
<td>Bronze 3/8&quot;-8 nut</td>
<td>McMaster-Carr</td>
<td>95072A124</td>
<td>1</td>
<td>24.96</td>
<td>24.96</td>
<td></td>
</tr>
<tr>
<td>Flexible shaft coupling hub, 3/8&quot;</td>
<td>McMaster-Carr</td>
<td>6408K9</td>
<td>1</td>
<td>2.95</td>
<td>2.95</td>
<td></td>
</tr>
<tr>
<td>Flexible shaft coupling hub, 1/8&quot;</td>
<td>McMaster-Carr</td>
<td>6408K9</td>
<td>1</td>
<td>2.95</td>
<td>2.95</td>
<td></td>
</tr>
<tr>
<td>Spider to connect coupling hubs</td>
<td>McMaster-Carr</td>
<td>6408K61</td>
<td>1</td>
<td>1.35</td>
<td>1.35</td>
<td></td>
</tr>
<tr>
<td>Ball bearings</td>
<td>McMaster-Carr</td>
<td>6384K23</td>
<td>2</td>
<td>8.38</td>
<td>16.76</td>
<td></td>
</tr>
<tr>
<td>3/8 x 3/4 Saft Collar</td>
<td>Menards</td>
<td>N/A</td>
<td>2</td>
<td>1.99</td>
<td>3.98</td>
<td></td>
</tr>
<tr>
<td>9/16-8 nut</td>
<td>Do it Best</td>
<td>N/A</td>
<td>1</td>
<td>1.49</td>
<td>1.49</td>
<td></td>
</tr>
<tr>
<td>1/2 Plastic Cup</td>
<td>St. Joe Place</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>10-32 0.50&quot; Screws and Nuts</td>
<td>Home Depot</td>
<td>N/A</td>
<td>3</td>
<td>0.25</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>#10 Washers</td>
<td>Home Depot</td>
<td>N/A</td>
<td>3</td>
<td>0.08</td>
<td>0.24</td>
<td></td>
</tr>
<tr>
<td>0.5&quot; HDPE</td>
<td>McMaster-Carr</td>
<td>8619K477</td>
<td>1</td>
<td>115.48</td>
<td>18.45</td>
<td></td>
</tr>
<tr>
<td>0.25&quot; HDPE</td>
<td>McMaster-Carr</td>
<td>8619K166</td>
<td>1</td>
<td>107.47</td>
<td>10.35</td>
<td></td>
</tr>
<tr>
<td>Status LED</td>
<td>University of Notre Dame</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Accelerometer</td>
<td>University of Notre Dame</td>
<td>N/A</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>70 durometer black epdm rubber bumper</td>
<td>Wefco Rubber</td>
<td>4134</td>
<td>6 ft</td>
<td>2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>ArbotiX Commander v2.0</td>
<td>ArbotiX</td>
<td>ASM-ARBX-CMD-V2</td>
<td>1</td>
<td>59.95</td>
<td>59.95</td>
<td></td>
</tr>
<tr>
<td>Arduino DUE</td>
<td>Arduino</td>
<td>RB-Ard-40</td>
<td>1</td>
<td>49.95</td>
<td>49.95</td>
<td></td>
</tr>
<tr>
<td>Arduino Mege Prototype Shield</td>
<td>DFRobot</td>
<td>DFR0016</td>
<td>1</td>
<td>6.80</td>
<td>6.80</td>
<td></td>
</tr>
<tr>
<td>XBee Module to DIP Adapter</td>
<td>NKC Electronics</td>
<td>SMD-0101</td>
<td>1</td>
<td>3.45</td>
<td>3.45</td>
<td></td>
</tr>
<tr>
<td>Sabertooth 2x25</td>
<td>Dimension Engineering</td>
<td>RB-Dim-23</td>
<td>1</td>
<td>124.99</td>
<td>124.99</td>
<td></td>
</tr>
<tr>
<td>ArbotiX Commander v2.0</td>
<td>ArbotiX</td>
<td>ASM-ARBX-CMD-V2</td>
<td>1</td>
<td>59.95</td>
<td>59.95</td>
<td></td>
</tr>
<tr>
<td>12 V 20 Amp NiMH Battery</td>
<td>Mega Batteries</td>
<td>17886</td>
<td>1</td>
<td>279.95</td>
<td>279.95</td>
<td></td>
</tr>
<tr>
<td>NiMH Battery; 7.2 V, 4500 mAh</td>
<td>Mega Batteries</td>
<td>14673</td>
<td>1</td>
<td>45.95</td>
<td>45.95</td>
<td></td>
</tr>
<tr>
<td>Energizer AA Batteries</td>
<td>Walmart</td>
<td>N/A</td>
<td>1</td>
<td>7.54</td>
<td>7.54</td>
<td></td>
</tr>
<tr>
<td>XBee 1mW Communication Module</td>
<td>Digi</td>
<td>PX-32405</td>
<td>2</td>
<td>21.95</td>
<td>43.90</td>
<td></td>
</tr>
</tbody>
</table>
Code - Receiver

#include <Commander.h>
#include <Sabertooth.h>

Commander Remote_Controller = Commander();
Sabertooth Drive_Motors = Sabertooth(128);

int Right_Motor = 2;
int Left_Motor = 1;
int Right_Speed = 0;
int Left_Speed = 0;
int Tackle_Pin_Input = 13;
// The setup routine
/*******************************************************************************/
void setup()
{
  // Set baud rate
  Serial.begin(9600);
  Sabertooth::autobaud(Serial);

  Serial1.begin(38400);

  pinMode(Tackle_Pin_Input, INPUT);
}
/*******************************************************************************/

// Loop routine
/*******************************************************************************/
void loop()
{
  if(digitalRead(Tackle_Pin_Input) == 0)
  {
    Tackled();
  }
  else
  {
    if(Remote_Controller.ReadMsgs() > 0)
    {
      MotorSpeed(-(Remote_Controller.walkV), Remote_Controller.lookV, Remote_Controller.buttons); //26 30
    }
  }
}
/*******************************************************************************/

//Sets all motor speeds to zero
/*******************************************************************************/
void Tackled()
{
  //Stop all wheels
  Drive_Motors.motor(Right_Motor, 0);
  delay(10);
  Drive_Motors.motor(Left_Motor, 0);
  delay(10);
}
/*******************************************************************************/

// Calculate the motor speeds and then apply them to the motors
/*******************************************************************************/
void MotorSpeed(int WalkV, int LookV, int Buttons)
{
    if ((Buttons&BUT_RT)>0)
    {
        Drive_Motors.motor(Right_Motor, .9*WalkV);
        delay(10);
        Drive_Motors.motor(Left_Motor, 1*LookV);
        delay(10);
    }
    else
    {
        //Implement the wheel speeds
        Drive_Motors.motor(Right_Motor, .36*WalkV);
        delay(10);
        Drive_Motors.motor(Left_Motor, .40*LookV);
        delay(10);
    }
}/*********************************************************************************/
// Include header files for the remote controller(Commander) and motor controllers(Sabertooth)
#include <Commander.h>
#include <Sabertooth.h>
#include <Servo.h>

Commander Remote_Controller = Commander();
Sabertooth Drive_Motors = Sabertooth(128);
Servo Hike_Servo;

int Right_Motor    = 1;
int Left_Motor     = 2;
int Right_Speed  = 0;
int Left_Speed   = 0;
int Tackle_Pin_Input = 13;
int Hike_Servo_Signal = 12;

int Servo_Pos = 900;

// The setup routine
void setup()
{
  // Set baud rate
  Serial.begin(9600);
  Sabertooth::autobaud(Serial);
  Serial1.begin(38400);
  pinMode(Tackle_Pin_Input, INPUT);
  Hike_Servo.attach(Hike_Servo_Signal);
}

// Loop routine
void loop()
{
  if(digitalRead(Tackle_Pin_Input) == 0)  // THIS ACCELEROMETER HAS AN INVERTED OUTPUT
  {
    Tackled();
  }
  else
  {
    if(Remote_Controller.ReadMsgs() > 0)
    {
      MotorSpeed(Remote_Controller.walkV, Remote_Controller.lookV);
      HikeBall(Remote_Controller.buttons);
    }
  }
}

// Sets all motor speeds to zero
void Tackled()
{
  Drive_Motors.motor(Right_Motor, 0);
delay(10);
Drive_Motors.motor(Left_Motor, 0);
delay(10);
}
****************************************************************************
/*============================================================================
void MotorSpeed(int WalkV, int LookV)
{
    //Implement the wheel speeds
    Drive_Motors.motor(Right_Motor, -.46*LookV);
delay(10);
    Drive_Motors.motor(Left_Motor, -.50*WalkV);
delay(10);
}
/*============================================================================
/*============================================================================
//Hike the ball to the Quarterback, engage servo
****************************************************************************
/*============================================================================
void HikeBall(int Buttons)
{
    if((Buttons&BUT_RT) > 0)
    {
        Servo_Pos = Servo_Pos + 10;
    }
    else
    {
        if((Buttons&BUT_LT) > 0)
        {
            Servo_Pos = Servo_Pos - 10;
        }
    }
    Hike_Servo.write(Servo_Pos);
}
/*============================================================================
/*============================================================================
Code - Quarterback

#include <Commander.h>
#include <Sabertooth.h>

Commander Remote_Controller = Commander();
Sabertooth Front_Motors = Sabertooth(128); // Bottom Motor Controller
Sabertooth Rear_Motors = Sabertooth(129);
Sabertooth Passing_Motors = Sabertooth(130);
Sabertooth Ball_Feed_Motor = Sabertooth(131); // Top Motor Controller

int Front_Right_Motor = 1;
int Front_Left_Motor = 2;
int Rear_Left_Motor = 1;
int Rear_Right_Motor = 2;
int Right_passing_motor = 1;
int Left_passing_motor = 2;
int Ball_Feed = 1;
int Front_Right_Speed = 0;
int Front_Left_Speed = 0;
int Rear_Right_Speed = 0;
int Rear_Left_Speed = 0;
int Passing_Motor_Power = 0;
int Ball_Feed_Speed = 0;
int Ball_Advanced_Input = 11;
int Ball_Home_Input = 12;
int Tackle_Pin_Input = 13;
int Power_RT = 0;
int Power_LT = 0;

// The setup routine
void setup()
{
  // Set baud rate
  Serial.begin(9600);
  Sabertooth::autobaud(Serial);

  Serial1.begin(38400);
  pinMode(Tackle_Pin_Input, INPUT);
  pinMode(Ball_Home_Input, INPUT);
  pinMode(Ball_Advanced_Input, INPUT);
  delay(1000);
}

// Loop routine
void loop()
{
  if(digitalRead(Tackle_Pin_Input) == 0)
  {
    Tackled();
  }
  else
if(Remote_Controller.ReadMsgs() > 0)
{
    MotorSpeed(Remote_Controller.walkH, Remote_Controller.walkV, Remote_Controller.lookH,
    Remote_Controller.lookV);
    MakePass(Remote_Controller.buttons);
    PassPower(Remote_Controller.buttons);
}

/*
********************************************************************************
************
********************************************************************************
//Sets all motor speeds to zero
********************************************************************************
************

void Tackled()
{
    //Stop all wheels
    Front_Motors.motor(Front_Right_Motor, 0);
    delay(10);
    Front_Motors.motor(Front_Left_Motor, 0);
    delay(10);
    Rear_Motors .motor(Rear_Right_Motor, 0);
    delay(10);
    Rear_Motors .motor(Rear_Left_Motor, 0);
    delay(10);
    //Stop passing pheels
    Passing_Motors.motor(Right_passing_motor, 0);
    delay(10);
    Passing_Motors.motor(Left_passing_motor, 0);
    delay(10);
}

/*
********************************************************************************
************
********************************************************************************
// Calculate the motor speeds and then apply them to the motors
********************************************************************************
************

void MotorSpeed(int WalkH, int WalkV, int LookH, int LookV)
{
    float boost;

    if (LookV > 74)
    {
        boost = map(LookV, 75, 128, 50, 100);
    }
    else
    {
        boost = 25;
    
}}
// Calculate the wheel speed
Front_Right_Speed = round((boost/100)*((WalkH + WalkV) + .75*LookH));
Front_Left_Speed  = round((boost/100)*((-WalkH + WalkV) - .75*LookH));
Rear_Right_Speed  = round((boost/100)*((WalkH + WalkV) - .75*LookH));
Rear_Left_Speed   = round((boost/100)*((-WalkH + WalkV) + .75*LookH));

// Implement the wheel speeds
Front_Motors.motor(Front_Right_Motor, Front_Right_Speed);
delay(10);
Front_Motors.motor(Front_Left_Motor, Front_Left_Speed);
delay(10);
Rear_Motors .motor(Rear_Right_Motor, Rear_Right_Speed);
delay(10);
Rear_Motors .motor(Rear_Left_Motor, Rear_Left_Speed);
delay(10);

// Power of the passing wheels is broken up to Slow, Intermediate and Fast
// L6 L5 L4 R3 R2 R1

void PassPower(int Buttons)
{
    if ((Buttons&BUT_R3)>0)
    {
        Passing_Motors.motor(Right_passing_motor, -40);// negative
        delay(10);
        Passing_Motors.motor(Left_passing_motor, 50);
        delay(10);
    }
    else 
    {
        if ((Buttons&BUT_R2)>0)
        {
            Passing_Motors.motor(Right_passing_motor, -60);
            delay(10);
            Passing_Motors.motor(Left_passing_motor, 75);
            delay(10);
        }
        else 
        {
            if ((Buttons&BUT_R1)>0)
            {
                Passing_Motors.motor(Right_passing_motor, -95);
                delay(10);
                Passing_Motors.motor(Left_passing_motor, 117);
                delay(10);
            }
        }
    }
}
delay(10);
Passing_Motors.motor(Left_passing_motor, 0);
delay(10);
}
}
}
}
}
/*----------------------------------------------------------------------------------------*/
//The right trigger will pass the ball
/*----------------------------------------------------------------------------------------*/
void MakePass(int Buttons)
{
    int Power_Push;

    if((Buttons&BUT_RT) > 0)
    {
        Power_Push = -30;
    }
    else
    {
        if((Buttons&BUT_LT) > 0)
        {
            Power_Push = 20;
        }
        else
        {
            Power_Push = 0;
        }
    }

    Ball_Feed_Motor.motor(Ball_Feed, Power_Push);
delay(10);
}/*----------------------------------------------------------------------------------------*/