Design, Communication, and Control of a Football Playing Robot Team

Team Members
Armela Mane, Raihan Mir, Jeremy Nyikos, Cliff Sidwell, Matthew Thompson, Colton Witte
Department of Engineering, Indiana University Purdue University Fort Wayne
Fort Wayne, IN 46818

Faculty Advisors
Guoping Wang, Zhuming Bi
Department of Engineering, Indiana University Purdue University Fort Wayne
Fort Wayne, IN 46818
wang@engr.ipfw.edu, biz@ipfw.edu

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 [2, 3]. 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 League. Due to time constraints and a limited team size, only three different robots of a complete robotic team are built. In this paper, the accomplishment through this project is summarized and the focuses are on the applications of IEEE standards for robot communication and control.**

Keywords—IEEE 802, Zigbee, Xbees, Wireless Communication, Playing Robot, Robot Team, Senior Design Projects

I. INTRODUCTION

Robots have an ever-growing influence on our daily lives. Robots are typical examples of mechatronic systems. Robotics related research and development is a type of ideal student projects due to the necessary application of Science, Technology, Engineering, and Mathematics (STEM) knowledge in creative engineering designs. To accelerate innovation in robotics, the 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. Due to the limitation of having just six team members, only three robots (also referred as players) were planned to be built during senior design project. The three players to be completed are the quarterback, receiver, and center. Those robots have been built using IEEE standards including those for wireless, software, and system engineering. These three design aspects are most critical to the execution of a football game and also possess the most challenging design opportunities. The completed players competed in the Collegiate Mechatronic Football game on April 19, 2013 and with very satisfactory performance.

A. Requirements and Specifications

A complete football robot team consists of eight robot players on the field. Based on the role a robot plays, football robots can be divided into offensive and defensive robots. Since offensive robots require more functions and capacities than defensive robots, 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. The quantified requirements of those robots are,

- 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, and
    - Remove power to drivetrain for 2 seconds.
- Robots are able to replace batteries within a minute.

Besides, each type of robot serves for a spefici role, for example,

- A center robot must deliver the ball to quarterback with a success rate of 75% within 20 seconds; it must be able to travel 50 feet within 8 seconds,
- A quarterback robot must complete a pass 65% of the time,
- A receiver robot must be able to travel 50 feet within 5 seconds,

B. Given Parameters

The following given parameters are specified by the organizer to ensure fairness of the contest.

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

C. 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.

Modular design concept is adopted to simplify the design process and maximize the flexibility of later design changes. The lists of hardware and software modules are as follows.

List of Hardware Modules:

- 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

List of Software Modules:

- Microcontroller
- Motor controller
- Programming Language
- Communication network and standards

D. Summary of Detailed Design

After a comparison of several design options for each modular component, the design concepts are finalized for these components as follows.

- The team decided to use the LeafLab Maple microcontroller. The features of a fast clock processor, high number of input/output pins and ability to use a Real Time Operating System made this board the best solution.
- The team decided not to use the Handheld Controllers previously used by Notre Dame. These are not as easy to use and troubleshoot. On the other hand, the Arbotix Commander V2.0 offers more programmable buttons and joysticks allowing a complete customization for each player. The capability of the controller to use Zigbee communication module to communicate with microcontroller made it a worthy match for the selected microcontroller.
- In order to meet the requirement of the robots to travel at least 10 ft/sec, two 4-7/8 inch diameter wheels, attached to a RS-540 motor are used. Due to the motors having a rated torque, BaneBots P60 Gearbox of 16:1 gear ratio is used.
- The locomotions of the center and receiver robots are similar; while the quarterback robot differs due to its functionality during the game. The drive trains of center and receiver use two driven wheels located in the center of the baseplate with 2 ball casters, one located in the front and one located in the back. On the other hand, the quarterback uses 4 omnidirectional wheels.
- In order to succeed in the ball transfer mechanism from center to quarterback, the alignment is achieved using a trapezoidal cutout section on the baseplate of the center; which mates with the complementing male end on the quarterback. Passing is achieved using a rotating clamp that positions the ball into the ball feeder of the quarterback.
- The ball feeder of the quarterback is mounted on a turn table at a 35^{o} angle. 2 passing wheels, equal to the ones used for the receiver and center drivetrain. Two RS-540 motors are used for the ball launching mechanism. Since a low torque is required, Banebots P60 Gearbox of 4:1 gear ratio is used.
- In order to maximize our catching capabilities, the cross pattern netting was implemented on the receiver.
- As an extra feature of the design, the team planned a simple tracking and positioning scheme using the vision system technique. The camera system chosen, CMUcam4, was mounted on the quarterback. Due to the time constraints, this feature was not functioned during the contest.

Three robots have been designed and tested within the budget. In the following sections, the building process has been overviewed and the robotic control as well as the implementation of wireless communication have been introduced.

II. Building Process

In this section, the mechanical building process is briefly described to show how robots were structured and programmed. The first stage of the building process was to build the chassis of the Center and Receiver. The baseplates were made of 0.5 inch thick High Density Polyethylene (HDPE) while the side panels and lids was made of 0.25 inch thick HDPE.

Firstly, the HDPE pieces for baseplates, lids and side panels were cut down from a large sheet of HDPE using a handsaw. 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. In a similar way, the slots for the wheels and motors were created. Wheels and gearboxes were then mounted. The side panels were attached to each

other using L-brackets and non-locking polypropylene draw latches were used on two sides to secure the lid on top of the side panels. A complete view of the Receiver chassis has shown in Figure 1.

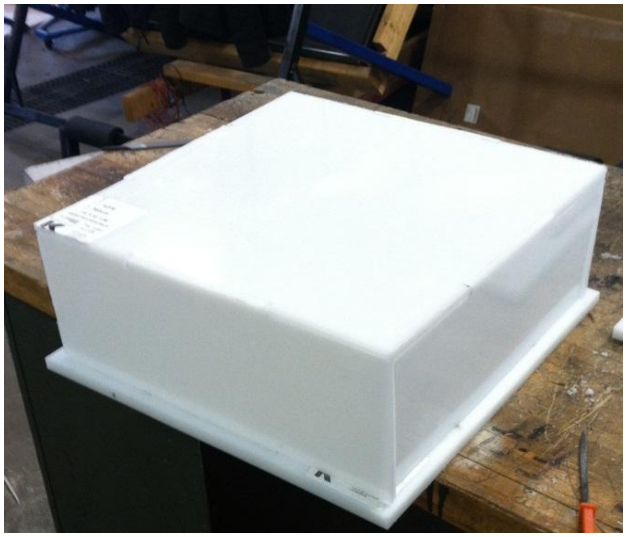

Figure 1. Receiver Assembled Chassis

Netting was then put on the Receiver to finish its mechanical assembly as seen in Figure 2.

Figure 2. Assembled Receiver.

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 was accomplished in Figure 3.

The Center delivers the ball to the Quarterback using a four bar linkage driven by a servo motor and given tension by a steel wire. The Quarterback delivers the ball to a set of pitching wheels using a lead screw delivery mechanism. Its final assembly can be seen in Figure 4.

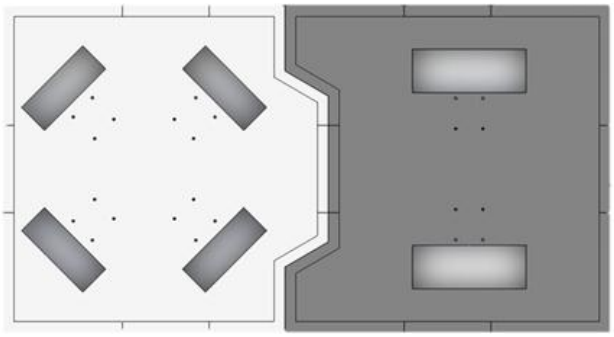

Figure 3. Center and Quarterback base in Solid Works

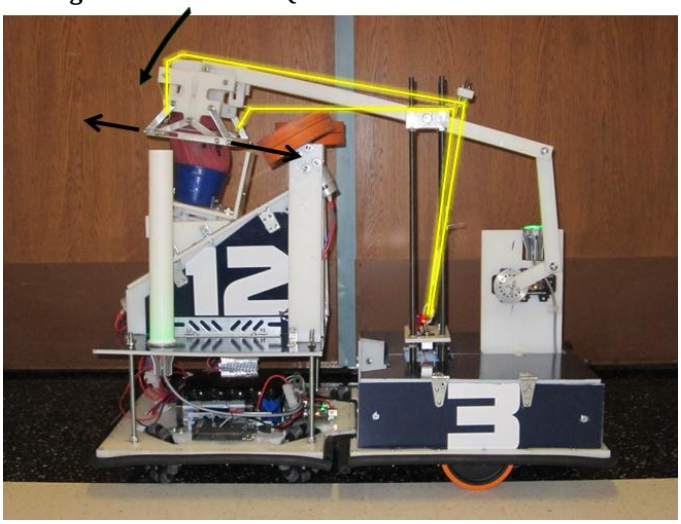

Figure 4. Delivery Mechanism

III. SOFTWARE DEVELOPMENT

The microcontroller codes are different from one player to another due to the differences of the capabilities and functionalities. The robot programs were developed using Sabertooth and Commander Libraries that were downloaded and added to the Arduino IDE.

A. Circuit Diagram

The circuit diagrams for the Receiver, Center, and Quarterback are shown in Figure 5, Figure 6, and Figure 7 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 (TX0) 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.

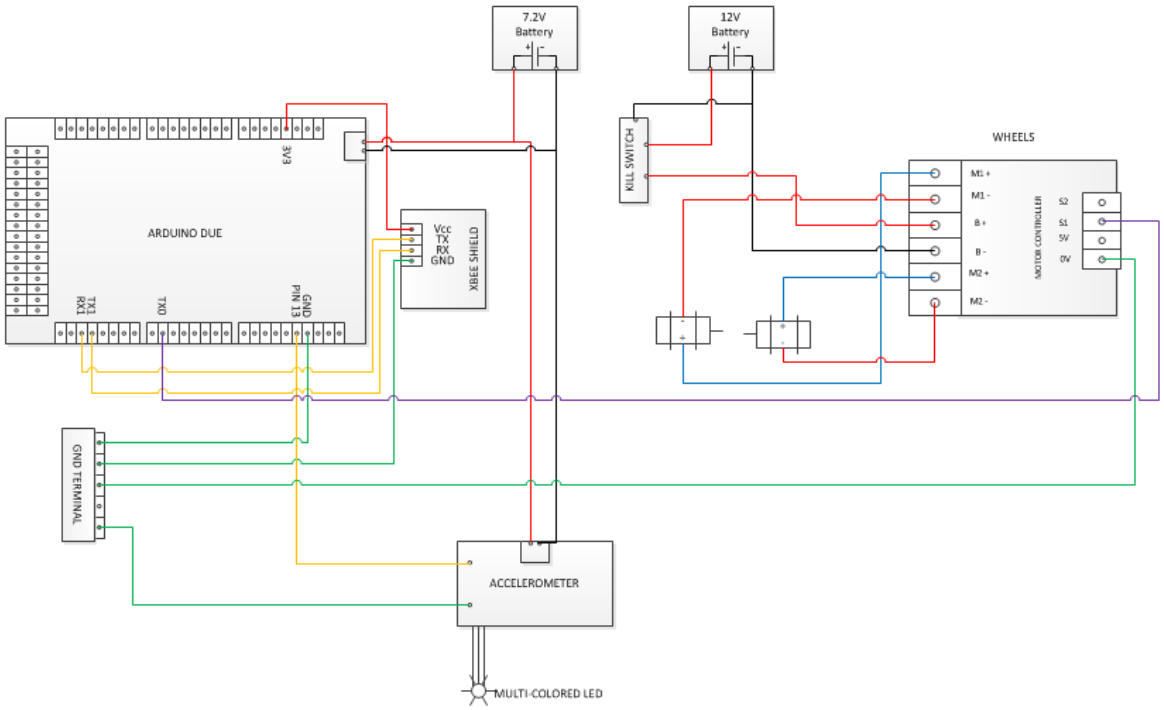

Figure 5. Circuit Diagram of Receiver Robot

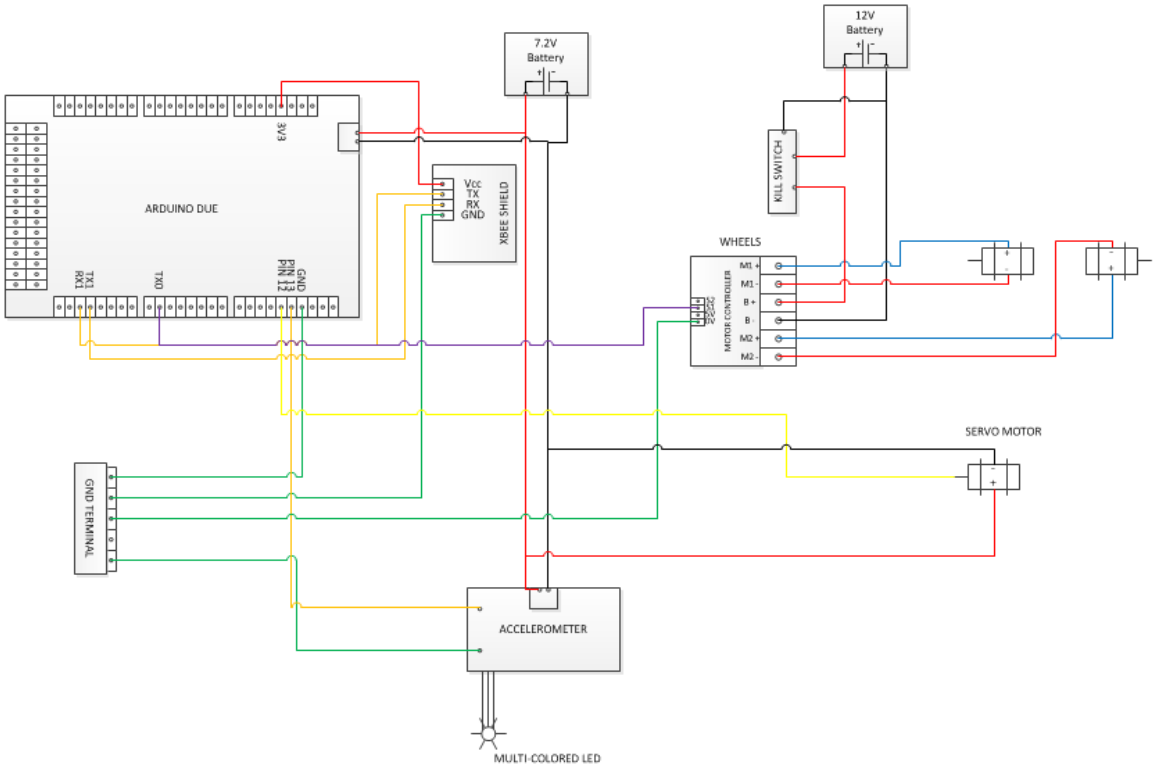

Figure 6. Circuit Diagram of Center Robot

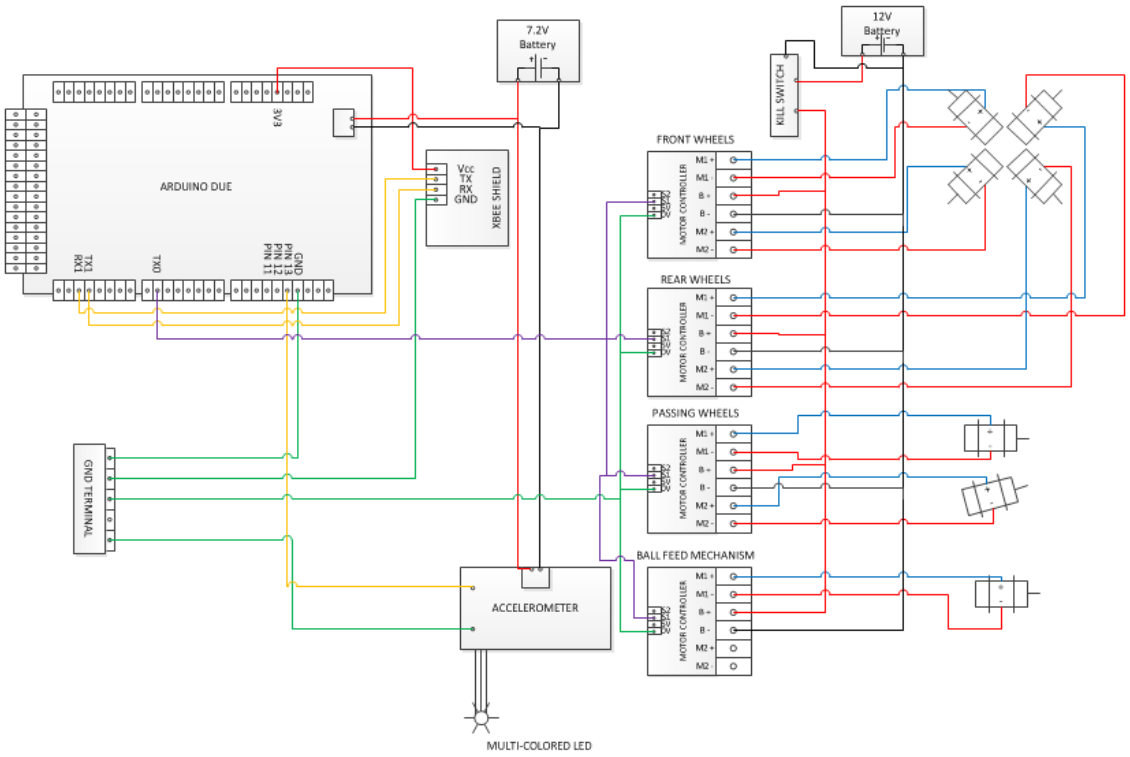

Figure 7. Circuit Diagram of Quarterback Robot

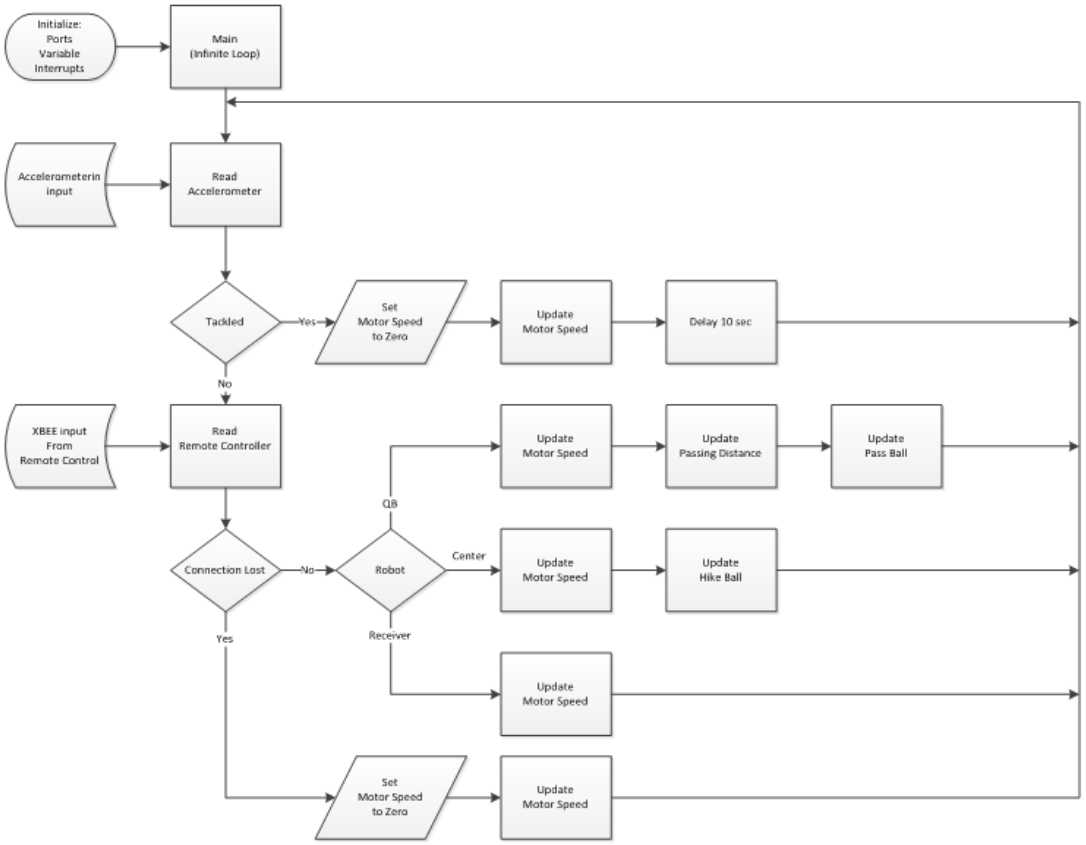

Figure 8. Program Flow Chart

B. Software Flow Diagram

As an example of software, the program flowchart of quarterback robot has been illustrated in Figure 8. The program 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.

Since the Center or Receiver performs much easier operations than the Quarterback, their programs can be implemented by simplifying the steps of the program for the Quarterback, which has not been covered in detail here.

IV. Communication and Control

The software used to program the electronic devices depends on the selected hardware. At 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) was that for a microcontroller. Generally, a microcontroller supplier recommended to use IDE.

The initial design was going to use an Arduino Uno since it has been widely recognized and 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.

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 on 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 came with its own IDE which 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 program robots.

As shown in Table 1, the first step was to identify all of the variables, signal wires, the inputs and outputs. 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.

Table 1. Quarterback Variables

Type	I/O or Internal Variable	Quarterback
Commander	Input	Remote Controller
Sabertooth	Output	Front Motors
Sabertooth	Output	Rear Motors
Sabertooth	Output	Passing Motors
Sabertooth	Output	Ball Feed Motor
Int	Output	Front Right Motor
Int	Output	Front Left Motor
Int	Output	Rear Right Motor
Int	Output	Rear Left Motor
Int	Output	Right Passing Motor
Int	Output	Left Passing Motor
Int	Output	Ball Feed
Int	Output	Front Right Speed
Int	Output	Front Left Speed
Int	Output	Rear Right Speed
Int	Output	Rear Left Speed
Int	Input	Tackle Pin input

The Xbee chips and the motor controllers were assigned a unique address which can be seen in Table 2. 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 2. Device Addresses

Device	Address
Xbee (x high light)	1111
Xbee (b high light)	2222
Xbee (e high light)	3333
Sabertooth (Receiver)	128
Sabertooth (Center)	128
Sabertooth (Quarterback)	128, 129, 130, 131

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 has its own 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 was 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 found. 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.

- Check if there is a new message on the UART
- Read the data from the UART
- The library handles how the message is broken up, for further details see the Commander's data sheet
- 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. 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

Old
Front_Right_Speed = round(.707*(WalkH + WalkV) + LookH));
Front_Left_Speed = round(707*(-WalkH + WalkV) - LookH));
Rear_Right_Speed = round(.707*(WalkH + WalkV) - LookH));
Rear_Left_Speed = round(.707*(- WalkH + WalkV) + LookH));
Current
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));

A layout of the Quarterback's handheld controller is shown in Figure 9. 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.

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 programed 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.

The robots use Sabertooth 2x25 motor controllers as seen in Figure 10. Each motor controller can regulate 2 motors at most. 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 Sabertooths 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 bye) can be ganged together on a single serial line.

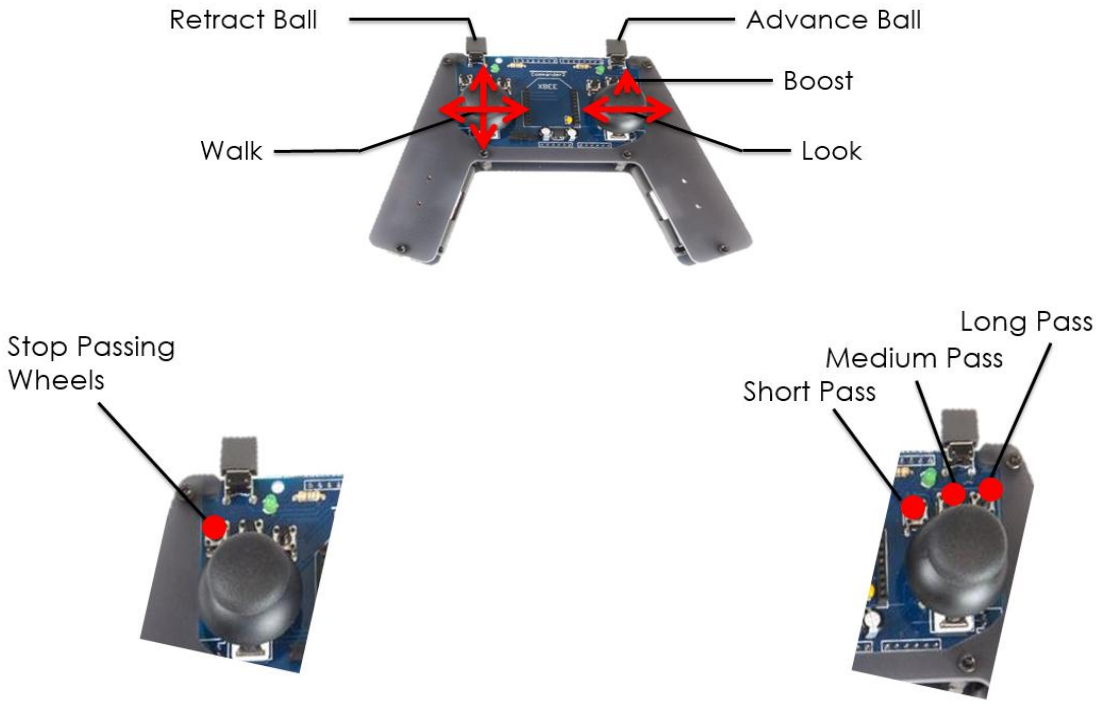

Figure 9. Quarterback trigger functions and drive train function

Figure 10. Sabertooth 2x25 V2.0

Figure 11. Microcontroller Assembly

V. Wireless Communications

IEEE 802.15.4 [4] was established in 2003. The mission for this standard is to empower the idea behind simple devices with a reliable, yet robust wireless technology that could run for years on standard batteries. One of the various protocols that are used to comply with this standard is Zigbee networking. IEEE 802.15.4 is a member of the IEEE 802 family, however that does not mean that all the feature of the IEEE 802 family of standards are involved. Some are not desired for this low-rate, low-duty cycle standard. Since this standard deals with mesh networking and sees Zigbee networking as an acceptable fit, it is easily understood why Xbee ZB chips were selected for wireless communication.

Now that IEEE 802.15.4 is understood, the Xbee chip can be analyzed. It features a 2.4 GHz frequency, 1.25/2 mW Power Output, 120 m range and a RF Data Rate of 250 Kbps. Since there are 3 robots to control at any given time, the three pairs need to communicate on different network ID's.

A MEGA prototype board is mounted on top of the Due for clean and secure wiring along with the Xbee chip. 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 soldered on the prototype board. A view of the entire microcontroller assembly can be seen in Figure 11.

While testing the Xbee chips, they could retain constant communication within 110 feet. If the signal is obstructed by going around a corner, than the length of constant communication is drastically reduced. However, this was not a concern seeing as how the robots need to function in an open area while playing.

VI. Conclusions

It is essential to ensure reliable wireless communication for the robot team. To test the wireless communication between a 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. It has been proven that the Xbee chip pair is able to retain constant communication within 110 feet.

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 while meeting IEEE standards for wireless, software, and systems engineering, specifically IEEE 802.15.4. 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.

Acknowledgements

We thank University of Notre Dame for sponsoring this project and IEEE for granting supplementary funds. In particular, we thank Notre Dame Professors, Dr. Michael Stanisic and Dr. James Schmiedeler, 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.

References

[1] G. Rosenthal, Notre Dame wins inaugural robot football game, http://www.nfl.com/news/story/09000d5d82a5defa/article/notre-dame-wins-inaugural-robot-football-game

[2] C. Szold, New technologies spread arrival of robots into our lives, http://www.usatoday.com/tech/news/story/2012-07-04/robotics-future/56022326/1

[3] IEEE standards: http://standards.ieee.org/findstds/index.html

[4] IEEE 802, http://en.wikipedia.org/wiki/IEEE_802.

[5] The rules of collegiate mechatronic football, Technical Report, Notre Dame University, 2012.

[6] “Holonomic Control of a robot with an omni-directional drive,” by Raul Rojas and Alexander Gloye Förster.