Design Document

Gateway

<04/17/12>

Coaches:
< Renée Andrews, Michael Ehrenfried, Bryan Rich >

Students:
< Teja Aluru, Lane Baumgarten, Peter Cooper, Rob Giometti, David Kading Shane Mathias, Russell Stewart >

Change History

Significant drafts, major edits, and structural changes will be listed here along with the initials of the editor and date.

- PMC 02/17/12 - First major draft completed for Pueblo tournament.
- PMC 02/24/12 - Design changes from Pueblo tournament added.
- PMC 02/29/12 - Edits completed per feedback from R. Andrews.
- PMC 03/02/12 - Second major draft completed for Colorado State Championships.
- SEM 04/17/12- Final draft completed for VEX Robotics World Championships- High School.

Table of Contents

Cover Page	1
Change History	2
Table of Contents	3
1.0 Introduction	4
Important Terms and Acronyms	4
2.0 Applicable Documents	4
3.0 Assumptions and Dependencies	5
4.0 Design Description	5
4.1 Design Overview and System Description	6
4.2 Functional Decomposition	6
4.3 Design Alternatives	7

1.0 Introduction

The Kent Denver Robotics Club participates in VEX competitions regularly as a challenge to improve our skills, and as an external measure of progress. We do this in accordance with our mission statement, “to learn about science, technology, engineering, and math through fun, innovative, and fulfilling team-oriented projects in robotics.” This year’s VEX Robotics competition, Gateway, is a game in which pairs of robots compete to place barrel and ball-shaped game objects into cylindrical goals of various heights.

This document details the process of designing the robot, including design alternatives, selection, and refinement. This is a working document and will continue to be expanded, updated, and revised throughout the design process.

Important Terms and Acronyms

BOM - Bill of Materials, an itemized list and price breakdown of the components used in building a system.
CVS - The Concurrent Versions System, a change management system.
SVN - Apache Subversion, a change management system maintained by the Apache Software Foundation.
AGMA - American Gear Manufacturers Association
COM - Center of Mass
COG - Center of Gravity

2.0 Applicable Documents

The following documents are important to the design process. They are used for background information, design research, and reference throughout the process. Numeric references refer to the corresponding document as defined in this section (i.e. “Document 2.1.2”).

2.1 VEX Gateway Official Documents
- **2.1.1** VEX Gateway Game Description and Scoring
- **2.1.2** VEX Gateway Manual
- **2.1.3** VEX Gateway - Appendix A Field Drawings, Specifications & BOM
- **2.1.4** VEX Gateway - Appendix B Robot Skills Challenge
- **2.1.5** VEX Gateway - Appendix C Programming Skills Challenge
- **2.1.6** VEX Gateway - Appendix D Inspection Guidelines
- **2.1.7** VEX Gateway - Appendix E Awards
- **2.1.8** VEX Gateway - Inspection Checklist
- **2.1.9** VEX Gateway - Referee's Scoresheet

2.2 Kent Denver Robotics Documents
- **2.2.1** Requirements Document

2.3 Official VEX Inventor's Guide

2.4 Kent Denver Robotics 2011-2012 Design Notebook

3.0 Assumptions and Dependencies

The following assumptions are made in the design of the robot and in all related documentation, unless otherwise stated.

3.1 Field

We assume the field on which the robot will perform is tournament-ready and complies fully with VEX rules (document 2.1.3.).

3.2 Parts

We assume all VEX parts and components perform as documented, within documented tolerances. References to component documentation are included as appropriate (document 2.3). Some vex parts include dramatic tolerances (eg field goals can be 30" +/- 1"); we will not design to the dramatics of these tolerances.

4.0 Design Description

This section presents the design of the robot as a whole. First, in section 4.1, we broadly describe the robot and explain its breakdown into subsystems. Section 4.2 specifically describes the functions for which each subsystem serves the robot and cross-references the specifications of the requirements document (2.2.1). Finally, section 4.3 describes the design alternatives considered in the design process and documents our selection and refinement of the final design.

The team chose to adopt an iterative design-build-test process in which we develop and implement a design, then test and revise it iteratively with thorough documentation made along the way. Revisions were then incorporated into the working design and the process was repeated. Throughout this process, we aimed to continuously make incremental improvements to a functional robot. We began our design process by breaking the project into subsystems, identifying the basic items our robot must accomplish, and drafting our requirements document (2.2.1), which has been updated continuously throughout the design process. After identifying preliminary requirements, the team brainstormed potential approaches to the problems each subsystem posed and recorded the process of refinement in the current document. We then evaluated our design alternatives based on agreed-upon criteria and selected the most suitable option for each subsystem. Next, the team divided into groups to refine and implement the selected design for a particular subsystem. Groups worked closely with one another to understand the interactions between subsystems. The final stage of building was integration, in which the groups came together and worked to smooth interactions between subsystems and transform the robot into a single, cohesive unit. We then thoroughly tested the robot, noting its strengths and weaknesses as well as any newly presented issues. Carrying forward information gained in testing, we revised our design and repeated the cycle.

The final design is centered around a conveyor belt like system for handling game objects, which is mounted to a four-bar mechanism. The geometry of the four-bar was carefully chosen as a compromise between forward reach and vertical lift, thus striking a compromise between scoring flexibility and avoidance of tipping.

4.1 Design Overview and System Description

For Gateway, we chose to pursue an aggressive tactic focused on scoring. Based on our previous experiences in tournament play and our interactions with other teams this competition season, we believe this to be the most effective strategy. Aside from the aggressive scoring robots, two significantly different designed bots appeared during the course of competition this year. A wall bot is a robot designed to expand to great widths as a means of limiting opposing robots from leaving or entering the isolation zone. Secondly, a "super-stacker" is a robot which gathers a large number of objects (generally in excess of 10) and places them on top of a goal. These two options are outlandish, can be easily defeated and require excessive design work to create a non-repeatable and non-reliable robot. Additionally, we reached the decision to build a aggressive scoring robot in accordance with the organizers' stated position on offensive tactics being most appropriate, as explained in VEX rule G11 (document 2.1.2).

For the purposes of design, the project has been divided into four subsystems: chassis and drivetrain, game object manipulation, sensors and code, and construction and rules. The following subsections describe the individual subsystems and their functions in more detail.

4.2 Functional Decomposition

The requirements traceability matrix, below, serves as a cross-reference to the requirements document (2.2.1). The functions of each subsystem are mapped to the corresponding requirements in the table below.

Requirements Traceability Matrix

Subsystem	Function	Requirement(s)
Chassis and Drivetrain	Mount parts of robot	3.1.6 - 3.1.13
	Frame and structure	3.1.14 - 3.1.15
	Drive the robot	3.1.2 - 3.1.5
	Comply with rules	3.1.1
Game Object Manipulation	Handle and score objects	3.2.1 - 3.2.7
Sensors and Code	Control robot	3.3.3 - 3.3.4
	Autonomous period	3.3.1

	Programming Skills	3.3.2
Construction and Rules	Comply with rules	3.4.1
	Build with good practices	3.4.2

4.2.1 Chassis and Drivetrain

The robot's chassis consists of the robot's basic frame structure and serves as the superstructure of the robot, to which the other systems mount. It also provides a stable, rigid base, on which, per their designs, the other systems rely. This system requires weight, material, stress, center of mass, rigidity and dimensional considerations.

The robot's drivetrain consists of motors, wheels, shafts, and other minor components, which develop or transfer mechanical power in order to move the entire robot. Analyses necessary for the drivetrain include gear stress (AGMA), shaft stress, power and torque output, gear ratios, and wheel size.

4.2.2 Game Object Manipulation

The robot must be capable of picking up, handling, and depositing game objects in order to score. The game object manipulation system is responsible for this and is the focus of much of our design effort. This system must be capable of handling both barrel and ball style game objects and of scoring these objects in any of the goals on the field. Additional challenges this system must overcome are posed by the tendency of the ball style game objects to roll, playing in crowded spaces, and barrels falling onto their sides.

It is of note that the bot can score in the corner goals without utilizing the object manipulation arm, but rather by pushing objects. This serves as a sort of redundant system and allows our robot to be a scoring threat even in the event of catastrophic arm failure or even simple, momentary current draw issues.

4.2.3 Sensors and Code

This subsystem consists of all the electronics on the robot, including the Cortex microcontroller, remote controls, sensors, and batteries. Additionally, this subsystem includes all code written to any other system components (the VEXnet joysticks and Cortex, in particular). Notably, this subsystem includes the autonomous period routine (including a Programming Skills Challenge routine) and the driver control scheme. Code and sensors are used to aid the driver and mechanical systems alike.

4.2.4 Construction and Rules

While not a physical part of the robot, this subsystem covers details of construction, compliance with rules, and best practices for the design of the robot. Since these details apply to the robot as a whole, we chose to handle them under a separate section, rather than integrating them into each subsystem.

4.3 Design Alternatives

This section presents each design alternative we considered in the process of designing the robot. We examined alternatives by subsystem, selecting the most suitable concept for each subsystem and refining it.

4.3.1 Chassis and Drivetrain

Chassis

The chassis is built primarily from wide C-channel steel and assembled in a “U” shape. We chose steel rather than aluminum for the enhanced stability offered by a heavier base. While vex does not specify materials or material properties, we assumed the steel was some form of basic carbon steel with an approximate density of 7.85 g/cm^2. Similarly, we assumed the density of the aluminum used was 2.7 g/cm^2. This dramatic difference in weight per unit volume in the chassis significantly helps to lower the robot's center of gravity. Additionally, similar aluminum parts supplied by Vex represent significant capital investment which was an unnecessary undertaking for our team. The broad use of C-channel is a result of the advantages it possesses in terms of moment of inertia when compared to other Vex standard parts. We opted for the horseshoe shape over the simpler rectangle to allow for greater clearance in “straddling” a goal. This shape also allowed us to mount both the microcontroller and batteries low and centered in our chassis, which further aids the COM and allows for simple microcontroller wiring.

Drivetrain

Over the course of the competition season, we have found the drivetrain to be the primary factor in limiting the speed of the robot, and with each revision we have improved the drivetrain to increase the speed of the robot without sacrificing stability.

Four Wheels

Initially, we built a square drivetrain of four standard 4” wheels with each wheel driven independently by a single 3-wire motor through a 60 tooth high strength gear. Although it was very reliable and stable, we found this design to be too slow.

In the next revision, we re-geared to use a 72:60 combination of spur gears. The new gear ratio helped to speed up the robot. We moved away from high strength gears both because we found them to be unnecessarily bulky and because 72-tooth gears are not available in high strength. This revision was an improvement over the previous configuration, but the robot was still sluggish.

Bevel Gears and Omnidirectional Wheels

Our next revision consisted of two major changes. First, we replaced the four standard wheels with three 4” omnidirectional wheels, allowing the robot to slide sideways in addition to normal driving and turning. Second, in order to obtain more clearance inside the horseshoe shape of the chassis, we moved the motors to the top of the chassis, perpendicular to the axles. This perpendicular orientation necessitated the use of bevel gears to transfer power from the

motors to the axles.

While this was our most maneuverable design yet, it was much less stable and tended to fall over, especially while carrying game objects. Additionally, though it did not happen during the short time in which we competed with this design, we feared another robot could easily knock ours down. With these concerns in mind, we returned to a rectangular drivetrain.

Chain and Sprockets

This symmetrical design consists of three inline 4” wheels on each side of the chassis driven by two motors via sprocket and chain. The middle wheel on each side is a standard wheel, while those in the front and back positions are omnidirectional wheels. While the omnidirectional wheels provide additional maneuverability, the standard wheels provide some resistance to sideways pushing by another robot.

Our latest revision has been to move up to 4” wheels as a means to gain higher top speed. While a range of gearing was tested, a 1:1.5 gear ratio proved to be the most reliable in terms of heat and current issues. We also invested in high strength chain kit in order limit the stress present in the regular chain and sprocket kit.

Vex publishes their plastics to be Delrin, a proprietary product produced by Dupont Chemical. Still, there are a variety of off the shelf versions of Delrin and even some custom versions. In considering the strength of the chain, we utilized the ultimate strength (the point at which the material will fracture and fail completely). There will certainly be a distribution of strengths as a natural result of the manufacturing process; we chose a conservative published value of 23 MPa. With an input torque of 27 in-lbs and a smallest cross sectional area of 0.0248 in^2, this resulted in an applied stress of roughly 15 MPa and a safety factor of about 1.5. We find this to be inline with the aerospace industry standard. Because aerospace must maximize safety without inducing significant risk, we consider a good standard for robotics. This analysis has proven itself in testing.

The purpose of all the chain stress analysis, as well as the gear ratio analysis was to provide us with an estimate of what parts we needed for the chain drive, as well as the knowledge of how fast our robot can travel for competition.

4.3.2 Game Object Manipulation

Much of our design effort went into the game object manipulation system because of its central importance in scoring. We initially considered five design alternatives for this subsystem and later revised and expanded them over the course of the year. This section addresses each of these alternatives and revisions.

Four-Bar Mechanism

This design consists of a handling apparatus mounted on the chassis via a four-bar linkage. The four-bar mechanism is actuated by motors which apply torque to the rear end of the linkage, raising the apparatus and extending it forward. The handling apparatus itself is a steel “box” which stacks up to four game objects vertically, picking up and retaining game objects using intake rollers mounted at the bottom of the box.

Four-Bar with Intake Mechanism

This design is similar to the four-bar system described above, with an additional mechanism to aid in picking up game objects. It uses a mechanism on top of the conveyor belt to aid in feeding objects, as well as dispensing objects into goals.

Four-Bar with Sliding Extension

Another variant on the basic four-bar system described above, this alternative adds a sliding mechanism to the top of the box, extending the box vertically and allowing it to hold three more game objects.

Conveyor Platform

In this design, two supports extend vertically from the chassis and mount turntable bearings. The turntables mount a platform, and driving them pitches the platform, which is free to rotate continuously in either direction. The platform mounts a conveyor belt like mechanism to handle game objects.

Split Conveyor

Similar to the conveyor platform design above, with the major difference being that the platform is broken into two smaller, independent units.

Evaluation of Design Alternatives

.

	Four-Bar	Four-Bar with Intake	Four-Bar with Extension	Conveyor Platform	Split Conveyor
Motors	7	7	5	10	3.5
Ease of implementation	8.5	7	5.5	9.5	3
Ease of scoring	7.5	7.5	7	8	7
Required sensing	9	9	7	9	6

Effectiveness of manipulating objects	8	9	9	7	8
Scoring ability (which goals?)	9	9.5	10	1	10
Weight	6	5.5	5	6.5	3
Ease of programming	7	7	6.5	7	5
Total	**62**	**61.5**	**55**	**58**	**45.5**

Six-Bar Mechanism

During our first two competitions (Thompson Robotics Expo and Thunder Ridge), we found that the four-bar mechanism was unable to lift the box apparatus high enough to score on the 30” goals and that it was not possible to adjust the geometry of the four-bar linkage to allow for a sufficient vertical range while remaining within the 18” initial size limit (requirement 3.4.1). In order to achieve both of these opposed objectives, we opted for a six-bar linkage. Although this does add some weight, we judged the ability to score on the 30” goals to be worth the trade.

Four-Bar Conveyor

At the Thunder Ridge competition, the box apparatus became tangled in a goal, rendering the robot unable to move for the duration of the round. We believed this problem was inherent in the design and likely to recur and therefore, reevaluated our design options.

This design retains the four-bar mechanism but replaces the “box” apparatus with a conveyor belt consisting of two sets of tank treads.

Four-Bar Conveyor with Sliding Extension

This adaptation of the four-bar conveyor design adds a rail-mounted extension to the back of the conveyor system which allows it to store two additional game objects.

Four-Bar Conveyor with “Shelf”

Similar to the sliding extension described above, this adaptation of the four-bar conveyor design adds a hinged extension to the back of the conveyor system which allows it to store two additional game objects.

4.3.3 Sensors and Code

Sensors

The robot employs two quadrature shaft encoders, five line-tracking sensors, and two potentiometers. The two shaft encoders are incorporated into the drivetrain and allow the robot to drive and turn precise amounts, by counting the amount of ticks the encoders; additionally, the encoders enable the robot to maintain a very straight line while driving forwards or backwards, which is particularly valuable in autonomous operation. We use five line-tracking sensors during autonomous operation to make use of the lines on the field to determine position and guide movement. We operate

Sensors are the basis for autonomous operation, providing the routine with the information necessary for decision making. However, the sensors are also of use during driver control.

Driver Control Scheme

The robot is controlled by a single driver using a VEXnet joystick. We elected to use a single driver rather than two because we do not believe the operation of the robot is complex enough to merit the additional difficulty which comes with a second driver. We also implemented multitasking with our drive code, allowing us to make different tasks for the drive, the arm, and our conveyor.

We apply high-pass filtering to the analog inputs from the joystick in order to address the problem of the VEXnet joysticks' analog sticks recentering imperfectly (failing to zero) when released.

Autonomous Routine and Programming Skills Challenge

During autonomous operation, the robot employs dead reckoning - a navigational technique in which the current location and heading are determined based on information about movement relative to a previous known location.

Though we implement autonomous routines for both interaction and isolation, we prefer to run in isolation. The interaction zone routine scores X points by Y.

In the Programming Skills Challenge (document 2.1.5), we do X.

For Programming Skills, our routine is based around line follow and dead reckoning code. Our robot will use the navigate method in order to travel quickly between waypoints, and will use line follow in order to align the robot to the center of a goal. The navigate method uses previously stored robot positions as well as basic trigonometry to travel in an orthogonal path toward the next specified location, and uses our quadrature encoders to check that the robot is traveling the specified path, and update the robot's position in the matrix. The line follow works by using a basic line follow routine with three sensors at the front of the robot, that adjusts the robots path should it stop following, as well as using two sensors in the back to check for intersections. Whenever the robot detects an intersection, it will return a certain case based on its position from jumpers, and have the robot either continue, turn left, or turn right.

Current Limiting

In order to mitigate the current draw issues, we made a separate task that constantly runs throughout the competition. The task makes a new motor array that constantly checks the acceleration of each motor on the robot. This ensures that the robot doesn't provide too much current to each motor within a short duration. This task also stops current flow when the robot motors are stalled.

Version Control

The code is managed using Git, which provides version control, change management, and project history. Git is widely used and accepted in professional software development, and projects such as the Linux kernel and the GNOME project employ Git as their sole system of revision control. In choosing a version control system, we identified three major options: CVS, SVN, and Git. We selected Git primarily because of its distributed nature, as we do not currently have a server available to us for development and the team works independently on different parts of the code. Git addresses both of these problems by allowing each client to own a complete project repository locally. Neither CVS nor SVN employs a similar architecture.

5.0 Detailed Design Description

5.1 Chassis and Drivetrain

Drivetrain

The drivetrain consists of two symmetrical "pods." Each pod is built of three inline wheels, driven by two high-power motors via chain and sprocket.

- Drivetrain
    - Sprocket and chain
    - Wheels
        - 4" wheels, four omni, four regular
    - Gear ratio on sprockets (1.5:1)
    - Two-wire motors
    - Shaft encoders geared down
    - Chain analysis
        - High strength chain between motors
        - Triple chain on wheels
    - Redundancy
- Chassis
    - Steel

5.2 Game Object Manipulation

- Four-bar
    - AGMA analysis on arm-lifting gears
    - Reliability analysis on four-bar height
    - Four-bar analysis: VLA, kinetics
- Conveyor intake

5.3 Sensors and Code

This subsystem is centered around the Cortex microcontroller and is responsible for controlling the robot during both remote-operated and autonomous operation.

5.3.1 Sensors

The robot uses sensors primarily in autonomous operation, during which they provide the information which enables dead reckoning (see 5.3.2).

Ultrasonic Sensors

The robot mounts three ultrasonic rangefinders.

Line-Following Sensors

We employ five line-following sensors in autonomous operation. Three of the sensors are mounted in a group at the front of the robot, spaced closely, while the remaining two are mounted at the center of the side members of the chassis.

Quadrature Shaft Encoders

Two quadrature shaft encoders are integrated into the drivetrain. They serve to measure the rotation of the components in the drivetrain, allowing the robot to move and turn precise amounts.

The shaft encoders are driven by a 10-tooth sprocket, putting them at a ratio of 1 : 2.4 to the wheels. This ratio increases the effective resolution of the shaft encoders from 90 to 216 ticks per revolution of the wheels, as one full turn of the wheels results in 2.4 turns of the shaft encoder.

5.3.2 Autonomous Operation

Dead Reckoning

We employ dead reckoning in autonomous operation. Dead reckoning is a navigational technique which relies on always knowing one's current location. We implement the technique by continuously keeping track of the robot's movement and, thus, its position relative to its starting point. Based on a known starting point, this relative location is translated into a location known in absolute space.

Autonomous Period

We implement an autonomous period routine for both red and blue alliances in both the isolation and interaction zones.

Programming Skills Challenge

We utilize line follow sensors as well as dead reckoning to complete our programming skills routine that is capable of scoring over 20 points

5.3.3 Driver Control

The robot is controlled by a single driver, although the rules permit two. We made this choice because we do not feel the operation of the robot is sufficiently complicated to warrant the additional time investment of practicing with two drivers.

5.3.4 Electrical Considerations

The Cortex microcontroller incorporates overcurrent protection into the motor outputs. Two separate time-dependent circuit breakers each protect half of the motors - motor ports 1-5 share one circuit and motor ports 6-10 share the other. Additionally, the motor outputs of the power expander module are protected by a third overcurrent device, identical to those of the microcontroller.

- Cortex diagram, showing motor wiring relative to circuit breakers

5.4 Construction and Rules