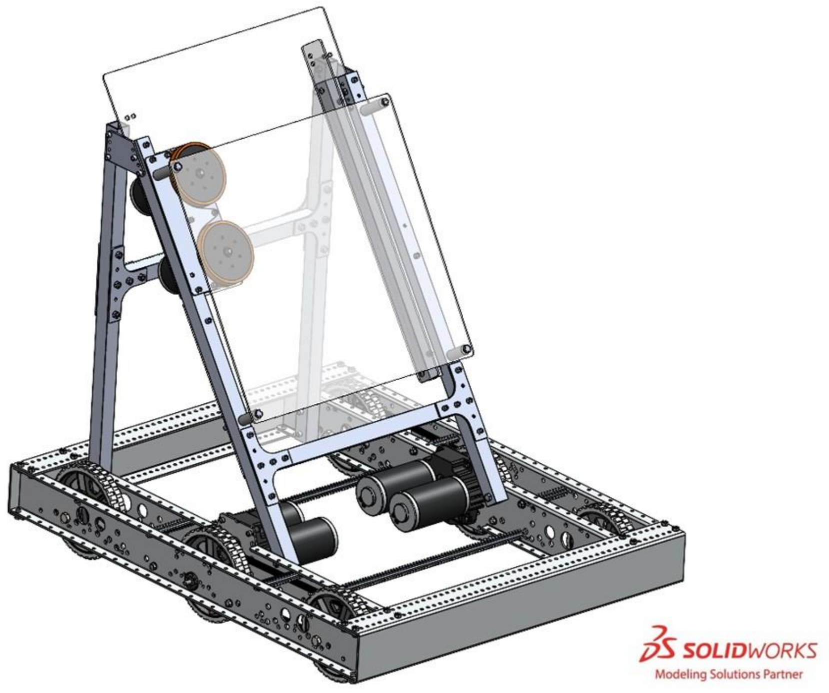
2024 *FIRST*® Robotics Competition KitBot Enhancement/Iteration Guide

Table of Contents

1 Introduction

Figure 1: 2024 KitBot



The KitBot for CRESCENDO℠ presented by Haas is capable of completing the following actions. Some actions will need the team to explicitly add code to make this possible (e.g. Auto code):

- Drive around the field (other than under the Stage) using a differential drivetrain (also commonly referred to as "tank") geared for a top achievable speed of ~15 feet per second (~4.5 m/s).
- Pre-load a Note for use in Auto
- Score Leave points
- Score Notes into the Speaker
- Collect Notes from the Source
- Play defense

This is a fairly basic set of capabilities with respect to all of the possible tasks in the game. Additionally, the KitBot has been designed to keep things very simple, which means there may be opportunities to iterate and improve on the existing capabilities it has. With this in mind, teams may choose to add additional components to allow the robot to pick game pieces up off the ground, climb on the Stage, and more! This document outlines a possible process you could follow to brainstorm and decide on possible improvements.

2 When to start thinking about changes

Either before or after you construct your KitBot you may wish to review the game and determine if there are additional capabilities you would like to try to add to your teams' robot. The KitBot is designed with future additions in mind, and we encourage most teams to put it together and play with it a bit before modifying or adding on, but there are potential benefits for considering additions before construction. These are some tradeoffs that may help you decide whether to do this first exercise before or after you have constructed the KitBot.

Before Constructing the KitBot

- Pros
    - Allows design members to work right away instead of having to wait until construction is complete
    - Easier to make changes to base design parts before they are assembled
- Cons
    - May be designing more robot than you can successfully build
    - May be making changes to parts of the base design you don't fully understand

After Constructing the KitBot

- Pros
    - Certainty that you have a robot that can play some aspects of the game
    - Better able to prioritize iteration (improving existing capabilities) vs enhancement (adding new capabilities)
    - Better understanding of function of base design and what can and can't change
    - Testing and practice can happen simultaneously with work on improvements
    - Software development improvements can be tested immediately
- Cons
    - Nothing to design until construction is complete
    - Changing base parts may require taking things apart
    - May want to use nuts and bolts instead of rivets to take things apart easier

3 Enhancement – Adding New Capabilities

This exercise can be completed before or after constructing the KitBot (see section 2 for pros/cons of each option). If completing this exercise after constructing the KitBot we recommended completing the brainstorming parts of all of the Enhancement and Iteration exercises (Sections 3 & 4) and then combining those to the down-selecting part of the exercise (Section 5) to prioritize what changes your team will pursue.

While not strictly required, having a set of team goals is very helpful for the narrowing portion of this exercise. For more information on setting team goals, see the "Goal Setting" video linked from the Team Management Resources page under the Team Organization section.

The exercises and techniques presented here comprise an abbreviated version of a complete game analysis and robot strategy process. For a more complete resource on this process, see the Strategic Design section of the "Effective *FIRST*® Strategies for Design & Competition presentation" from the Technical Resources page under the Other Technical Resources->Scouting/Strategy section.

3.1 Analyzing the Game

3.1.1 Analyzing the game – What else could our robot do?

The first step of brainstorming potential enhancements is to analyze the game and come up with a list of possible things the robot could do. An abbreviated set of questions for doing this is provided below. For a more complete process, check out the "Kickoff Worksheet" on the Technical Resources page under the Other Technical Resources->Kickoff section. This is different than just a list of game tasks (though that's a good place to start!). Don't start thinking about how the robot will do these tasks yet. Below is a list of a questions that can help guide your team in coming up with this list:

1. What are the ways to score points in the game?
2. Can any of these items be broken down further (e.g., scoring from different distances may be a unique capability)?
3. How does the robot acquire game pieces? Often there are multiple ways to acquire that require unique capabilities.
4. Are there any other tasks that might help your alliance (e.g., field elements to manipulate, ways to help partners, etc.)?
5. Are there ways to slow the opponents scoring down that require unique robot capabilities (i.e. not just defending using the robot drivetrain)?

Looking for an example? Check out Appendix A – Example Basic Game Analysis, 2023 for a possible task list for the 2023 game.

3.1.2 Analyzing the game – How do we rank high?

If your team goals involve trying to rank as high as possible (e.g. trying to be an alliance captain, trying to maximize District Points to qualify for a District Championship, etc.), it's important to review the criteria teams are ranked by. If your team goals don't involve ranking high (e.g. be picked for an alliance, design a cool mechanism, win an award, etc.) you can skip this section. The criteria for ranking are typically found in the Tournament section of the manual and are usually based on Ranking Points (RPs), earned for Wins or Ties and for completing specific game objectives. Assuming a typical *FIRST*® Robotics Competition game with ranking in this style, ask yourself these questions:

1. Other than Winning Matches, how can Alliances earn Ranking Points?
2. For each of these RPs what robot capabilities are needed to:
    a. Maximize our ability for our alliances to earn the RP?
    b. Achieve the RP if playing with 2 other robots like ours?
    c. Make the minimum contribution towards the RP for our alliance?
3. What are the 1st and 2nd tiebreakers after RP (it's typically unlikely to have a tie beyond this and very unlikely for such a tie to decide more than a rank or two at most)? These should not be a major influence in selecting capabilities but may provide a small push in the case of a tough decision between two capabilities when prioritizing.

3.1.3 Analyzing the game – Winning matches?

If your team goals involve winning matches, you'll need to spend some time looking at the game and thinking about what a winning alliance looks like. Depending on your exact goals, some of these sample questions may be more or less relevant.

1. List out all of the scoring tasks in the game with point values.
    a. How many times an alliance can complete each of these tasks? This may not be a single clean answer, some games may have tasks that are mutually exclusive.
2. For each task, estimate how fast you think the task can be accomplished.
    a. The answers to this will vary highly based on the strength of the robot, you likely want to pick a particular "class" of robot and answer consistently with that level in mind. For scoring that requires a game piece, remember to include the time to go get that game piece and return to the scoring location.
3. For some tasks, you may also want to consider a "success rate". This is what percentage of the time you think the robot will complete the task in the time you have estimated.
    a. For a launching task this might be obvious, how often will the game piece go in the goal. For other tasks it may be less obvious but try to consider how often you may have to re-align and try again for example. Express the answer as a decimal or fraction (e.g. 80% success rate = .8)
4. Use the numbers from the previous three steps to calculate the points/second for each scoring task (success*points/seconds) and sort from most to least efficient.

Based on the timing and efficiency list you came up with, evaluate the following (skip any that are not appropriate for your team goals):

1. What does the match look like for an average winning alliance in Qualifications?
2. What does the match look like for an alliance that would win about 80% of Qualification matches?
3. What does the match look like for an alliance that can win a Playoff Match?
4. What does the match look like for an alliance that wins the event?

Using your teams' goals and an assessment of your capabilities, estimate your role in each of these alliances:

1. How many points are you trying to contribute to the alliance?
2. Are there specific tasks in these alliances that you think you need to contribute?
3. Are there specific tasks you want to leave for alliance partners to contribute?

3.2 Identifying Enhancements

Now compare the capabilities of the KitBot to the list of tasks and point totals you just generated based on your team goals. What capabilities do you need to add in order to meet the points and capabilities identified? Which of these is most important?

3.3 Next steps

If you're considering enhancements before you construct your KitBot, jump down to Section 5 for some ideas about how to turn these ideas into a plan. If you've already built your KitBot, proceed to Section 4 for information about how to purposefully test your robot and identify other potential improvements to include in your planning process.

4 Iteration – Improving Existing Capability

Iteration is a core component of an engineering design process and designing a *FIRST* Robotics Competition robot is no different. Once you've built your KitBot, you'll want to use any time remaining before or between your events to do two things: practice and iterate. For more information about driver selection and practice, see "Guide to Selecting Drivers" and "Improving Driver Performance" on the Technical Resources page under the Other Technical Resources section.

A simple process for iteration is outlined below consisting of the following steps:

- Test
- Ideate
- Plan
- Implement

4.1 Purposeful testing

The first step in an iteration process is to identify areas for improvement. For your KitBot we recommend doing this via purposeful testing. The majority of this testing will involve using your robot as you expect to during a match, which will also serve as driver practice. The one exception is testing to failure. Provided you have enough time (we recommend at least a week) before your event, you can intentionally test your robot in compromising situations such as running into things aggressively, acquiring too many game pieces, pushing buttons in an incorrect order, etc. to see what breaks. As you test the robot, write down answers to the following questions:

1. What broke? How did it break?
2. What is the robot doing inconsistently (e.g., failing to acquire game pieces, dropping game pieces, missing scoring opportunities, etc.)?
3. What tasks are taking the drivers a long time to complete (e.g., acquiring game pieces, driving across the field, lining up to score, releasing the game piece, etc.)?

If you have more than one game piece available, you may want to make some careful comparison of robot performance with each game piece to see if there is any variation. Then take a game piece or two and change them in ways you expect might happen during gameplay (over or underinflated, distorted, scuffed, or torn, etc.). Repeat the robot testing and document any changes in behavior.

4.2 Brainstorming improvements

Now that you have identified potential areas for improvement, the next step is to generate ideas to improve those weaknesses. This document breaks the brainstorming up into two pieces, mechanical and software/electrical. This is done to help make the process a little more manageable, and because these items often require different sets of resources, they can frequently be pursued in parallel.

4.2.1 Brainstorming improvements – Mechanical

For each of the areas of potential improvement identified in your testing, brainstorm some potential mechanical changes that could improve performance. Some prompting questions for each of the three categories are included below:

4.2.1.1 What broke?

1. Are there ways we could change the geometry of the part that broke to make it stronger?
2. Are there alternate materials we could use that would strengthen the part with the same geometry?
    a. Remember that stronger does not always mean stiffer, sometimes making a part out of a more flexible material (e.g. changing an aluminum part to polycarbonate) may allow it to absorb energy without reaching permanent deformation.
3. Do we actually want the part to be stronger, or do we want to make spares?
    a. If the cause of the failure seems abnormal and may happen infrequently in a tournament, if at all, you may want to leave the part as is, or sometimes even make it slightly weaker. While this may seem counterintuitive at first, designing in a known weak link and being prepared to replace it if damage occurs can help prevent unexpected damage to more expensive or more difficult to replace parts. Mechanical shear pins and electrical fuses are some examples of such sacrificial parts.

4.2.1.2 What is inconsistent?

1. Are there ways we could tweak part geometry to try to improve consistency?
    a. Ideas for this often include adding or removing constraints or guides, increasing or decreasing compression on a game piece, removing gaps or crevices a game piece may be inadvertently interacting with, or adding additional wheels or rollers connected to an existing system.
2. Are there ways we could change the materials to try to improve consistency?
    a. Ideas for this are typically looking to increase or reduce either flexibility or friction.
3. Are there ways adding additional actuation could increase consistency?
    a. Examples of this include actively expanding a mechanism for acquiring game pieces, adding additional latches or gates in a game piece motion path, adding additional actuation to help ensure game piece orientation in a system, etc.

4.2.1.3 What takes a long time?

1. Are there mechanical guides that can be added to speed up the driver completing this task?
    a. Examples include wedges or funnels that interact with field elements, or a physical marker on a robot that helps the driver better sight alignment with a field element.
2. Are there ways to make the robot more robust to misalignment?

    a. Examples include increasing the opening size for game piece acquisition or changing the geometry of scoring to work from different alignments.
3. Is the issue how long the mechanism takes to complete the action? Can it be sped up by changing gearing?
    a. If you plan to speed a mechanism up, make sure to check how much current it is drawing in its current configuration. Current will scale approximately linearly with gearing so if speeding the mechanism up will push the current to concerning levels, you may need to add another motor as well.

4.2.1.4 Specific ideas for the 2024 KitBot

Below are some specific suggestions for potential mechanical improvements for the 2024 KitBot. Skip this section if you'd prefer to stick with the ideas that you generate yourself.

1. **Add shielding to prevent inadvertent Note possession** - The KitBot has locations where a Note could get stuck inadvertently (due to incorrect lineup when acquiring, rebound off the Speaker, etc.) which would count as your one allowable Note to control. Shielding could be added to these locations to reduce this likelihood.
2. **Close ring acquisition gap** - To keep things simple, the KitBot acquires Notes with a completely passive ramp. This leaves a gap between the ramp and the wall caused by the ramp needing to stop at the edge of the Frame Perimeter. An actuated mechanism could be used to fill this gap after the start of the match, extending the ramp out over the bumpers. You likely want to stop at or slightly before the edge of the Bumper to allow the Bumper to continue to absorb any impacts with the wall.
3. **Reduce launcher spin-up**. – The launcher on the KitBot needs a second to spin up before being ready to launch. You could reduce this time by increasing the motor power applied to the launching wheel by changing to a more powerful motor or adding a second motor to power the wheel. You could also mitigate this with a software change by changing the controls to allow your driver to spin up while the robot is still getting into position and then launch only when fully ready.

4.2.2 Brainstorming improvements – Electrical/Software

Electrical and software improvements are grouped together as they often go hand in hand. Making software improvements to a robot often requires sensor input to allow the software to react to what is happening. Just like mechanical brainstorming, go through each of your areas of potential improvement and brainstorm how software feedback or automation could provide improvement.

4.2.2.1 What broke?

1. Could the failure have been prevented by travel limits for a mechanism?
    a. These are most often implemented using limit switches to denote the edges of allowable travel. An alternative is using "soft limits" for mechanisms that already have

absolute position feedback (i.e. you know your mechanism is at 30 degrees so you can limit it with software from traveling further in that direction).

2. Could the failure have been prevented with current limits?
    a. Current limits can help reduce the maximum force a mechanism can output and can limit motor overheating if a motor becomes stalled due to an unexpected mechanism jam.
3. Could the failure have been prevented by software interlock?
    a. Sometimes robots have mechanisms that should only be actuated when the robot is in a certain state otherwise something might break. These exclusions can be enforced in the software to help prevent the drivers from making a mistake and damaging the robot.

4.2.2.2 What is inconsistent?

1. Are the controls intuitive and obvious?
    a. If the drivers are making mistakes, it could mean that they need more practice, but it also could mean that the controls aren't intuitive. Once you've selected your drivers, make sure to work with them to ensure the controls make sense to them. What might be obvious to one person might be weird to another. Bad control design can also include using a single button to toggle a state back and forth that the drivers can't see. For example, if pressing a single button turned the launcher wheel on and pressing it again turns it off, how do you know if you pressed it and remembered to turn it off?
2. Are there timing issues that software automation could help prevent?
    a. Sometimes inconsistency comes from a driver doing something at the wrong time. Automating this timing with software can help by removing the human inconsistency from the process. An example of this is the Note launcher. In order to work effectively, the front wheel has to be spun up for a bit before the back wheel pushes the Note forward. Rather than have the drivers try to time this, the provided software handles this timing automatically by controlling the whole sequence with a single button press-and-hold.
3. Are there sequences that could be handled by software automatically?
    a. Sometimes to perform an action a robot has to actuate multiple mechanisms in a specific order, with or without specific timing. Can software automation be used to either enforce this order or automate the whole sequence to prevent error?
4. Are there software interlocks that could help prevent issues?
    a. Inconsistency can sometimes come from performing an action when the robot wasn't in the right state (not aligned with the field, interacting mechanism not properly prepared, etc.). Can software be used to detect any of these incorrect states and prevent the driver from performing the action?
5. Does robot behavior seem inconsistent even with what looks like correct driver actions?
    a. Sometimes this inconsistency comes from changes in the robot or environment such as a different battery voltage or a little more friction as a part wears. Adding sensors and software controls can help ensure the robot does the same thing every time.

4.2.2.3 What takes a long time?

1. Can alignment be automated?
    a. The most common way is to use vision sensing to detect the robot's location on the field and use that navigate a robot to a desired location, but depending on the task other sensors such as rangefinders or beam breaks may be useful as well.
2. Can you give the drivers more information?
    a. The drivers can do their best to use their eyes to see what is going on, but the robot can also help by sending additional signals. Streaming a camera to the driver station, displaying robot states on the dashboard, using lights on the robot to signal state, and adding controller feedback (if applicable) are all examples of ways to provide information about robot state back to the drivers.

4.2.2.4 Specific ideas for the 2024 KitBot

Below are some specific suggestions for potential software automation for the 2024 KitBot. Skip this section if you'd prefer to stick with ideas you generate yourself.

1. **Closed loop launcher control** – By either switching the motor powering the front launcher wheel to a motor with a built-in encoder (Venom, NEO/Vortex, Falcon, Kraken) or by adding an encoder directly to or geared off of the existing motor, you can switch from the current open loop control (estimating wheel speed based on provided voltage and time) to closed loop control (precisely controlling wheel speed based on feedback). To get velocity information from your chosen sensor, you will need to consult the documentation for the appropriate motor or sensor. To use this information to control wheel speed, see the Advanced Controls Introduction section of the WPILib documentation, especially the article on "Tuning a Flywheel Velocity Controller."
2. **Note detection** – A switch, proximity sensor, or break beam sensor could be added to the robot to detect when a Note is in the robot. You could use this information to automatically stop running the wheels inward to reduce motor heating and could provide driver feedback via lights, dashboard, or controller rumble (if using a controller with rumble such as an Xbox controller). This may also reduce the likelihood of acquiring multiple Notes at the same time.
3. **Vision alignment** – The KitBot can be easily aligned with the Speaker due to the large opening, proximity to your drivers, and ability to run the Bumpers up against the Subwoofer to align the depth. Aligning with the Source may be more difficult due to the distance from your drivers and the narrower width of the slots on the Source compared to the Speaker. Adding a camera to process AprilTags can be used to align with either of the two outside slots on the Source. You can do this by adding a webcam plugged into the roboRIO or by adding an external vision system like a Limelight, RaspberryPi with WPILibPi or co-processor board with PhotonVision. Once you have data from the AprilTag, you can use it to align the robot to the Source by:
    a. Having the driver position the robot near the Source and pointing in approximately the right direction such that you're sure the tag is in view.
    b. Having a driver press and hold a button to initiate vision targeting.

c. While the button is held, control the rotation of the drivetrain using the location of the tag. You can likely use simple proportional control to get close enough. Compare the location of the tag to the center of the image, multiply the result by some constant that you tune and apply the result to the rotation of the drivetrain (apply the result as the turn value of arcade drive function or half of the value in opposite signs to each side of a tank drive function).
d. Allow the driver to keep control of the propulsion component of the drivetrain by either continuing to apply the joystick value to the forward/back movement of an arcade drive function, or by adding the joystick value to the AprilTag value for a tank drive (for tank drive you will also have to detect if either value exceeds 1 and scale both sides down if it does).
e. The driver can then drive the robot towards the Source while the robot keeps itself pointed at the tag. When the Bumpers reach the Source wall, the center of the robot should be approximately aligned with the center of the slot. The robot may not yet be aligned flush against the wall. Continue pushing the robot towards the Source wall and the Bumpers should square up against the wall. You can also potentially release the alignment button and apply a little turn in the appropriate direction to ease this part of the process.

A similar process could be used to align with the Speaker. Or, for more advanced use, you could use the AprilTags to get complete position information for your robot and align or plan a path anywhere on the field!

4.3 Next Steps

If you considered enhancements before constructing your robot, decide whether you want to revisit brainstorming on that topic to add to your pool of iterations, or whether you have a list of leftover ideas to reconsider. If you haven't considered enhancements yet, collect your full list of ideas from both exercises and proceed to Section 5 for tips on how to prioritize them.

5 Down-selecting – Turning ideas into a plan

If you're considering enhancements before constructing your KitBot, you may not have any ideas for iteration yet. Proceed through this exercise with just the enhancements. You can always come back and do it again with remaining enhancement ideas plus new iteration ideas after you've built and tested your robot. If you've already constructed your robot, we recommend combining the ideas for new capabilities with the ideas for iteration into one down-selection exercise.

Turning your list of possible improvements into a prioritized list to act on is admittedly a difficult exercise that involves weighing many factors. There are many ways to approach balancing the perceived benefit of each change against the difficulty of implementing it and what team resources may be required to implement it. One possible method is detailed below.

5.1 Prioritizing

Leaning on your analysis of the game, and your vision of your robots' role in alliances, prioritize your potential improvements and enhancements from most impactful to least impactful. Don't worry yet about what is easy and what is difficult, or what uses what resources, just create an ordered "dream list" of robot improvements. An example is provided below:

1. Fancy new robot capability
2. Auto thing that helps with RP
3. Endgame thing that helps with RP
4. Tweak to improve scoring accuracy
5. Make spares of component X

5.2 Analyzing – Estimating Resources and Scope

In this section, you are going to look at your list and assess the resources required to execute each improvement. Rather than just assigning one concept or number with the difficulty of an idea, try creating a couple different categories that represent the various resources that may be required. Some of these categories may represent physical resource limitations such as material on hand, machine availability, or budget to purchase new components/material and some may represent more abstract resources like the total time/capability of students and mentors in a specific aspect of the team.

One possible set of categories is: Design, Manufacturing, Software, Materials/Cost. For each idea, assess the resource cost for each of these categories and assign a value 0-5 (feel free to substitute whatever scale you wish) with 0 being no resources required and 5 being a very significant resource requirement. For priority, use the rank your created from Step 5.1, 1 through N (it's ok if these are on a different scale than the category values).

For a new mechanism, ideally you could come up with ballpark numbers without needing to brainstorm and select what specific mechanism you would use to complete the task, but for some teams this may be difficult. You may need to take a detour at this point to brainstorm or prototype specific mechanism ideas in order to assess their resource requirements, see Section 5.5 for additional resources.

A game and robot agnostic example is provided in Table 1.

Table 1: Improvement Resources Example

Improvement	Priority	Design	Manufacturing	Software	Materials/Cost
Fancy new robot capability	1	5	3	3	3
Auto thing that helps with RP	2	0	1	3	0
Endgame thing that helps with RP	3	3	2	1	1
Tweak to improve scoring accuracy	4	1	3	0	1
Make spares of component X	5	0	2	0	2

5.3 Analyzing – Set your "budget"

Now that you have set relative resource costs of your ideas, you need to establish a "budget" describing your team's capabilities in each category. For each of your categories, using your assigned costs as a reference for scale, consider your team resources in the time between now and your event. Make sure to consider the time you will need just before the event to test and practice with any improvements and pack your equipment for the event. Assign an overall budget for each category that represents your resources available in that category between now and your next event.

As an example, let's consider a team with 2 CAD students and a CAD mentor, only 1-2 software students with no software mentor, a company that helps with manufacturing parts in addition to the equipment at the team shop, and a substantial stock of raw materials and COTS parts. This team might describe their budget for these categories as:

- Design – 5
- Software – 3
- Manufacturing – 10
- Materials - 10

5.4 Making plans

Now that you have a budget and some costs, it's time to put them together into a plan of what ideas to pursue. One approach is to simply go down the priority list until the budget in any category is exhausted, then continue down adding any items that only affect categories with remaining budget.

Another more complex approach is to have individuals or small groups put together sample plans and then evaluate these plans as a team and select what the group believes will result in the best robot. This can sometimes result in more optimal use of resources.

Let's revisit our example table above with our theoretical team who has set their Design resource budget at 5 points (we're going to ignore their budget in other categories for simplicity). Using the first approach, this team would choose to add a "Fancy new capability", exhausting their whole design budget, and then "Make some spares of component X".

Table 2: Example Improvement Plan – Option 1

Improvement	**Priority**	**Design**	**Manufacturing**	**Software**	**Materials/Cost**
Fancy new robot capability	1	5	3	3	3
Make spares of component X	5	0	2	0	2

This plan makes two improvements and uses 5 points of manufacturing resources, 3 of software and 5 of materials.

Using the second approach, someone may suggest instead that the team skip that first improvement and spread their design budget points across the next three improvements.

Table 3: Example Improvement Plan – Option 2

Improvement	**Priority**	**Design**	**Manufacturing**	**Software**	**Materials/Cost**
Auto thing that helps with RP	2	1	1	3	0
Endgame thing that helps with RP	3	3	2	1	1
Tweak to improve scoring accuracy	4	1	3	0	1
Make spares of component X	5	0	2	0	2

While this plan misses out on the item with the highest individual priority, it makes 4 improvements instead of two and may better utilize the available manufacturing (8 vs 5) and software (4 vs 1) resources. The team would then have to weigh the value of these 4 improvements vs the 2 from the other plan and decide which approach will produce the best robot to meet the team's goals.

5.5 Next Steps

5.5.1 Enhancements – New Mechanisms

For adding new mechanisms, the next step is to start brainstorming possible implementations and then prototyping to determine a direction. The following resources from the Technical Resources page, under Mechanical Resources->General, may be a helpful place to start:

- "Prototyping 101"
- "Prototyping Worksheet"
- "Mechanisms Worksheet"
- "NASA RAP Robotics Design Guide"

5.5.2 Enhancements – Software Capabilities

For adding new software capabilities, check out the "Control System and Programming Documentation" as well as the "Programming 101" from the Software/Electrical Resources->Software section of the Technical Resources page.

5.5.3 Project Plan

The last thing you likely want to do before jumping in and getting started on the work is sketch out a rough project plan. This can be as simple or as detailed as you want, but you likely want at least a few milestones between now and your next event. These will help you check if you are on track and allow you to add resources (such as meeting time) or reduce features before you reach a point where you are showing up at the event with an unfinished or untested robot. Cutting a feature or two to make sure you have time to test, and practice will likely pay off in the end.

6 Appendix A – Example Basic Game Analysis, 2023

6.1 Example - Robot Task List

Below is an example robot task list (see Section 3.1.1) for the 2023 *FIRST* Robotics Competition game CHARGED UP^{SM} presented by Haas. This list was made in hindsight, individual teams may have had more or fewer tasks on their own lists.

Ways for a Robot to score:

- Cone high
- Cone mid
- Cone low
- Cube high
- Cube mid
- Cube low
- Balance Bridge

Further breakdown of scoring:

- Balance Bridge
    - Balance by driving on the bridge from the wide edge
    - Balance from the narrow side of an already balanced bridge

Ways to acquire game pieces:

- Double substation
    - Cubes
    - Tipped Cones
    - Upright Cones
- Single Substation direct
    - Cubes
    - Tipped Cones
    - Upright Cones
- Floor
    - Cubes
    - Tipped Cones
        - Point first
        - Flange first
    - Upright Cones

Acquiring game pieces from the opposite side of the robot from where they are scored may speed up paths, especially important in Auto (may matter less for drivetrains that can rotate while moving such as mecanum or swerve).

Other tasks:

- Feeding game pieces to partners?

Defensive tasks:

- Acquiring opponent game pieces? (Ability to acquire with no extension to legally grab pieces from opponents LOADING ZONE)

6.2 Example - How to Rank High

Below is an example of how to rank high in the 2023 *FIRST* Robotics Competition game CHARGED UP presented by Haas.

1. Other than Winning Matches, how can Alliances earn Ranking Points?
   a. Link RP – Score at least 5 Links or at least 4 Links with Co-Op (this refers to the original Kickoff value, not the modified Championship value)
   b. Charge Station RP – Score at least 26 charge station points
2. For each of these RPs what robot capabilities are needed to
   a. Maximize our ability for our alliances to earn the RP?
      i. Score 15 total game pieces mixed across both types in at least the lower and mid row to create 5 Links
      ii. Balance the Charge Station in Auto so only a 2 Robot balance is needed in the endgame to secure the RP
   b. Achieve the RP if playing with 2 other robots like ours?
      i. Score ~5 total game pieces mixed across both types in at least the lower and mid row to help create 5 Links
      ii. Balance the Charge Station in Auto so only a 2 Robot balance is needed in the endgame to secure the RP
   c. Make the minimum contribution towards the RP for our alliance?
      i. Score ~3-5 total game pieces in at least the lower row to contribute towards Links.
      ii. Help balance the Charge Station in the End Game to contribute towards Charge Station points.
3. What are the 1st and 2nd tiebreakers after RP (it's typically unlikely to have a tie beyond this and very unlikely for such a tie to decide more than a rank or two at most)? These should not be a major influence in selecting capabilities, but may provide a small push in the case of a tough decision between two capabilities when prioritizing.
   a. Charge Station points

b. Auto points

6.3 Example - How to Win Matches

Below is an example points/time analysis for the 2023 game. To simplify slightly, this analysis looks at the Teleop period only and ignores Autonomous. It also assumes that scoring game pieces are contributing to Links which may not be a valid assumption if choosing a Cube only strategy. The timing estimates and success rate used are approximations and represent a robot trying to make playoffs (~50th percentile) at an average regional or district event.

Table 4: Example Points Over Time Analysis

Scoring Action	**Points**	**Time**	**Success Rate**	**Pts/Second**
Low Cube	2 + 5/3 (Link)	22	95	.158
Low Cone	2 + 5/3 (Link)	30	95	.116
Mid Cube	3 + 5/3 (Link)	27	90	.122
Mid Cone	3 + 5/3 (Link)	35	85	.088
High Cube	5 + 5/3 (Link)	30	85	.160
High Cone	5 + 5/3 (Link)	38	80	.119
Balance 1 Robot	10	15	90	.6
Balance 2 Robots	20	20	85	.85
Balance 3 Robots	30	25	80	.96

Next, some approximate answers to the questions about winning alliances.

1. What does the match look like for an average winning alliance in Qualifications?
   3x Mobility = 9 points
   Auto Docked and Engaged = 10 points
   3x Auto game pieces = +3 Points
   2x Low Links = 2*5 + 6*2 = 22 Points
   2x High Links = 2*5 + 6*5 = 40 Points
   Docked and Engaged x2 Robots = 20 Points
   Park = 2 Points
   Total = 106 Points

2. What does the match look like for an alliance that would win about 80% of Qualification matches?

   3x Mobility = 9 points
   Auto Docked and Engaged = 10 points
   4x Auto game pieces = +4 Points
   3x Low Links = 3*5 + 9*2 = 33 Points
   2x High Links = 2*5 + 6*5 = 40 Points
   Docked and Engaged x2 Robots = 20 Points
   Park = 2 Points
   Total = 118 Points

3. What does the match look like for an alliance that can win a Playoff Match?

   3x Mobility = 9 points
   Auto Docked and Engaged = 10 points
   4x Auto game pieces = +4 Points
   3x Low Links = 3*5 + 9*2 = 33 Points
   2x High Links = 2*5 + 6*5 = 40 Points
   Docked and Engaged x2 Robots = 20 Points
   Park = 2 Points
   Total = 118 Points

4. What does the match look like for an alliance that wins the event?
   3x Mobility = 9 points
   Auto Docked and Engaged = 10 points
   5x Auto game pieces = +5 Points
   3x Low Links = 3*5 + 9*2 = 33 Points
   1x Mid Link = 5 + 3*3 = 14 Points
   3x High Links = 3*5 + 9*5 = 60 Points
   Docked and Engaged x3 Robots = 30 Points
   Park = 2 Points
   Total = 162 Points

Finally, using a team that is trying to make playoffs as a goal, some approximate answers to what the team should be trying to contribute:

1. How many points are you trying to contribute to the alliance?
   ~30-40 points
2. Are there specific tasks in these alliances that you think you need to contribute?
   Dock and engage in Auto
3. Are there specific tasks you want to leave for alliance partners to contribute?
   No

6.4 Example - Enhancements

Let’s assume that the 2023 KitBot was designed to deliver Low Cubes only. Looking at the point total and tasks identified above, adding a Dock and Engage Autonomous routine would allow the robot to meet the tasks and point totals identified. Looking at the points/time table, the next capability to consider adding would probably be Mid Cubes. This is almost as efficient as the low cubes (and could be better if it can be done as quickly as Low Cubes) and gives the team 3 more scoring opportunities.