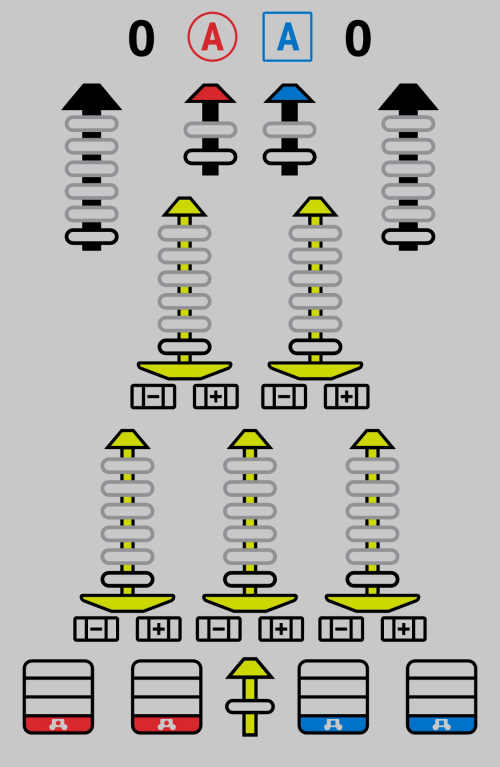
Entries By Date

Entries By Project

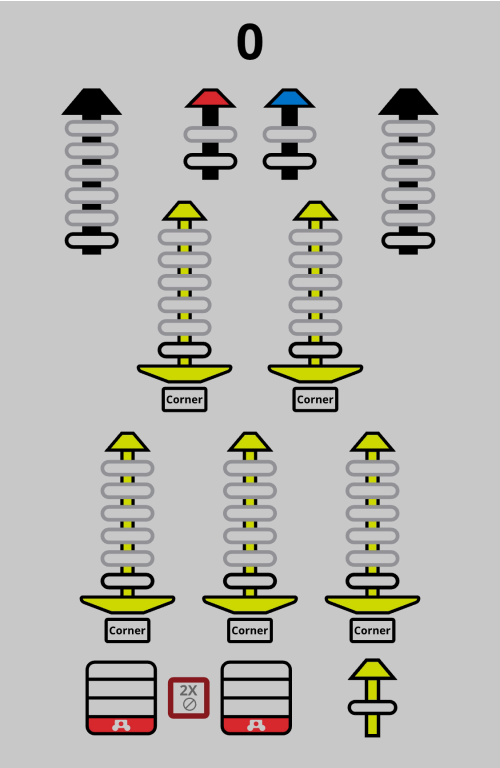
V5RC Hub Scoring Calculator Issues Focus: **Scoring Calculator**

By: **Zoe Rizzo** Date: **05/15/2024**

We need some way to score matches and skills to plan strategies and determine what is the most important scoring elements during matches. The official V5RC Hub mobile app has a scoring calculator, but has some major issues surrounding the game manual rules and usability.



V5RC Hub VURC Match Scoring Calculator



V5RC Hub VURC Skills Scoring Calculator

Game Manual Inconsistencies

The scoring calculator has some inconsistencies with the game manual.

Alliance Stake Rings

You are unable to put wrong color rings on alliance stakes in the scoring calculator. While alliance wall stakes are protected so that only that alliance can interact with their stake, there is still the possibility of a ring of the wrong color ending up on a wall stake. A ring of the wrong color will not score any points, but will count towards the maximum ring limit of 2.

<SG10> Alliance Wall Stakes are protected.

If a Ring of the opposing color ends the Match in a Scored position on an Alliance Wall Stake, that Ring should not be considered as Scored, and will not earn points for either Alliance (it will still, however, count toward the maximum number of 2 Rings that can be placed on an Alliance Wall Stake).

Mobile Goals in Corners

The scoring calculator allows more than two mobile stakes to have the +/- modifiers. In matches, only one mobile stake can be placed in each corner, meaning two mobile stakes can have the + modifier, and two mobile stakes can have the - modifier.

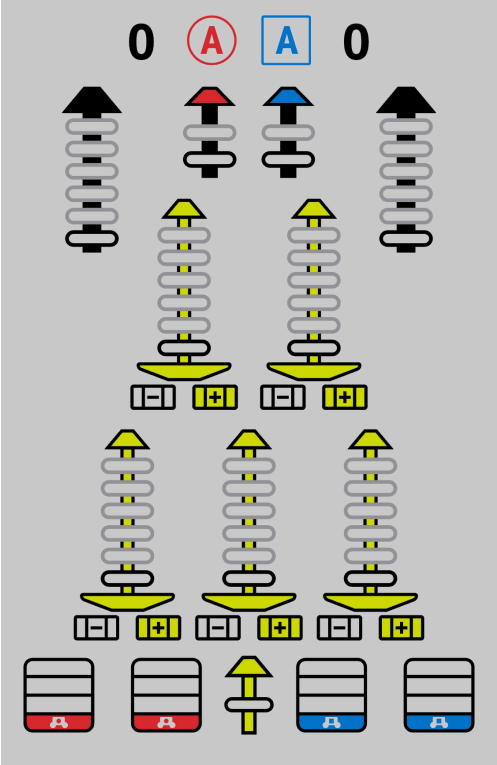
<SC5> Only one Mobile Goal may be considered Placed in each Corner.

If multiple Mobile Goals meet the above requirements in the same Corner, the following criteria will be used as a series of “tiebreakers” to determine which Mobile Goal is Placed.

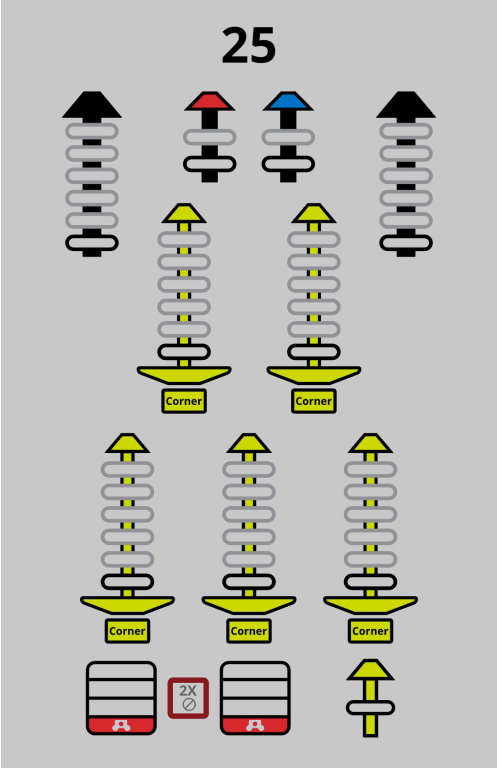
The same goes for the VURC match scoring calculator. Only one mobile stake can be placed in each corner, allowing for a maximum of four mobile stakes in corners. However, all five mobile stakes can be selected as "in a corner."

<RSC7b> Each Mobile Goal Placed in a Corner will receive 5 points.

Rule <SC5> and its note still apply, and only one Mobile Goal may be Placed in each Corner.



5 mobile stakes in positive corners in match scoring



5 mobile stakes in corners in skills scoring

Match Autonomous

Match autonomous is set to 0-0 unless specified otherwise in the scoring calculator. This is incorrect as the autonomous bonus is 3 points for each team if it is a tie.

<SC2b> Scoring of the Autonomous Bonus is evaluated immediately after the Autonomous Period ends.

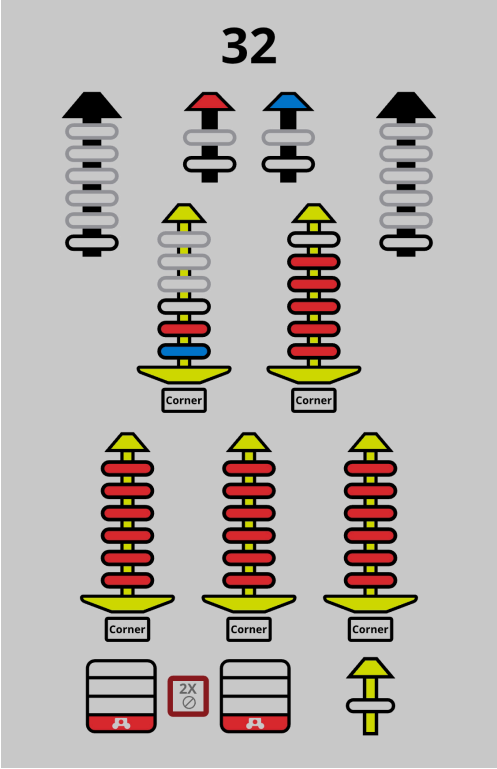
If the Autonomous Period ends in a tie, including a zero-to-zero tie, each Alliance will receive an Autonomous Bonus of three (3) points.

VURC Skills Red Rings

The scoring calculator scores red rings that are placed on top of blue rings as legal. However, any red rings placed on top of blue rings are worth 0 points, as well as the blue rings. This should also invalidate all blue rings, since, for blue rings to be scored, all red rings have to have a point value.

<RSC5> Any red Ring Scored above a blue Ring on the same Stake will not have a point value.

<VURS4a> Each blue Ring only has a point value if all red Rings in the Match have been Scored on Stakes and have point values.



The scenario pictured above should be worth 31 points -- 4 red rings worth 3 points, 19 red rings worth 1 point, and 1 red and 1 blue ring worth 0 points. However, it shows 32 points, meaning it counts either the last red ring or the blue ring as 1 point, which is incorrect.

Scoring UI Issues

The scoring calculator also has some usability issues. These issues are not a necessity to fix, but would be nice to see. Some of these issues include not being able to see how many rings are left to score or displaying a score breakdown to see where points come from.

The V5RC Hub App Scoring Calculator is not sufficient to use for strategies, scoring throughout the season, etc. We need a better way to score matches and skills this season.

Continued in **Google Sheets Scoring Calculator** (Pg. 10)

Google Sheets Scoring Calculator

Focus: **Scoring Calculator**

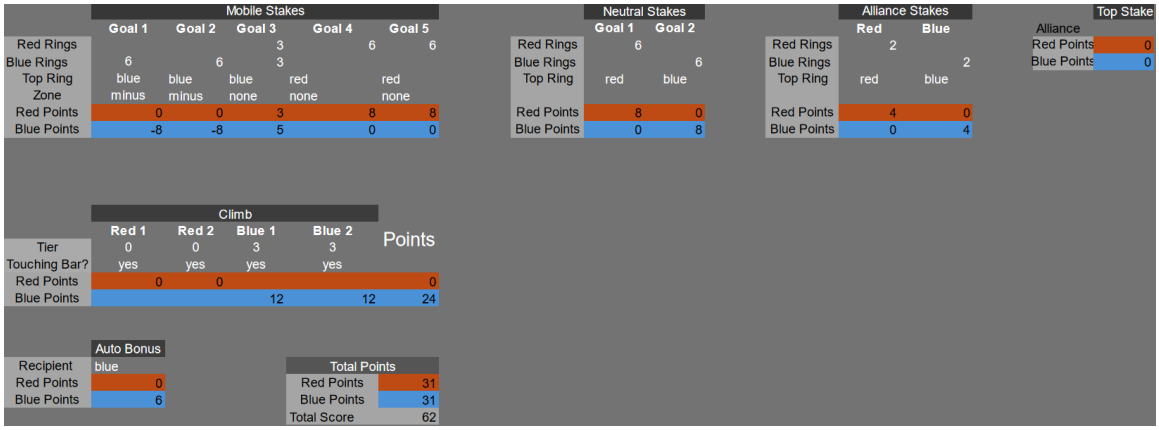
By: **Zoe Rizzo**

Date: **05/20/2024**

Continued from **V5RC Hub Scoring Calculator Issues** (Pg. 4)

As an alternative to the V5RC Hub scoring calculator, we created a scoring calculator on Google Sheets.

This allows us to see score breakdown by showing exactly how many points each team is getting from each stake, climb, and autonomous.



Mobile Stakes	Goal 1	Goal 2	Goal 3	Goal 4	Goal 5
Red Rings			3	6	6
Blue Rings	6	6	3		
Top Ring	blue	blue	blue	red	red
Zone	minus	minus	none	none	none
Red Points	0	0	3	8	8
Blue Points	-8	-8	5	0	0

Neutral Stakes	Goal 1	Goal 2
Red Rings	6	
Blue Rings		6
Top Ring	red	blue
Red Points	8	0
Blue Points	0	8

Alliance Stakes	Red	Blue
Red Rings	2	
Blue Rings		2
Top Ring	red	blue
Red Points	4	0
Blue Points	0	4

Top Stake	Alliance
Red Points	0
Blue Points	0

Climb	Red 1	Red 2	Blue 1	Blue 2	Points
Tier	0	0	3	3	
Touching Bar?	yes	yes	yes	yes	
Red Points	0	0			0
Blue Points			12	12	24

Auto Bonus	
Recipient	blue
Red Points	0
Blue Points	6

Total Points	
Red Points	31
Blue Points	31
Total Score	62

There are two main issues with this scoring calculator: usability and skills scoring. The usability issue is not a major issue, it just means that scoring matches can be a bit unintuitive compared to something like the V5RC Hub app. There would be no easy way to "clear" the field and reset all values. Also, having everything listed out instead of displaying graphics can be unappealing and difficult to navigate.

However, the bigger issue is with skills scoring. Since skills has so many complex rules, especially surrounding the blue rings, it is very difficult to create a completely accurate skills scoring calculator. Most of the rules that would make this difficult surround ring position - for example, a red ring has to be scored somewhere below a blue ring for the blue ring to earn points. A red ring also cannot be scored above a blue ring. This would be very difficult to program using Google Sheets, and would likely be easier to do in Python or JavaScript.

While this solution would work for match scoring, it is pretty unintuitive and does not have an option for skills.

Continued in **GitHub Pages Scoring Website** (Pg. 11)

GitHub Pages Scoring Website

Focus: **Scoring Calculator**

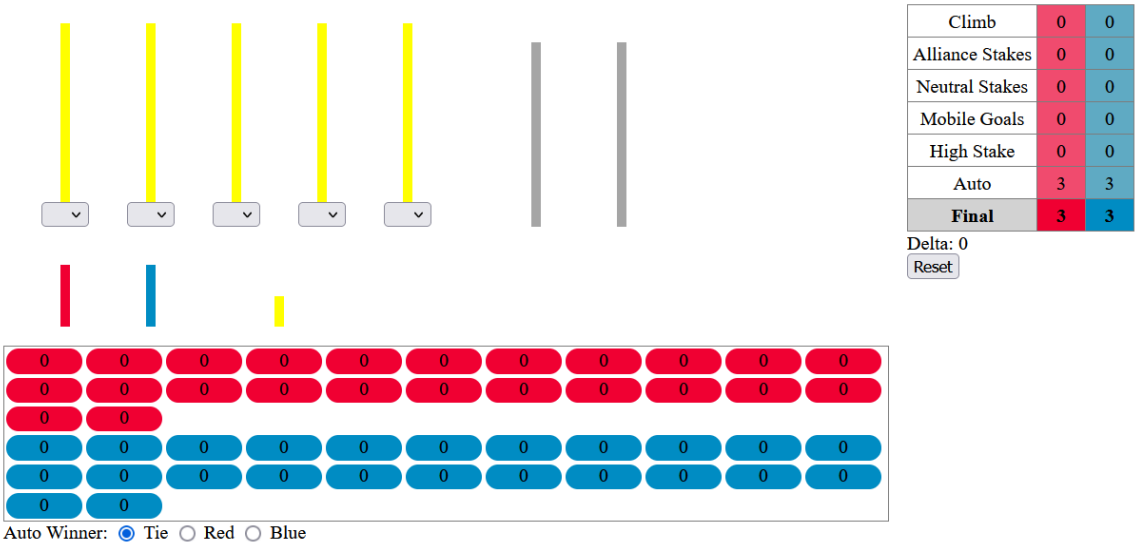
By: **Zoe Rizzo**

Date: **05/22/2024**

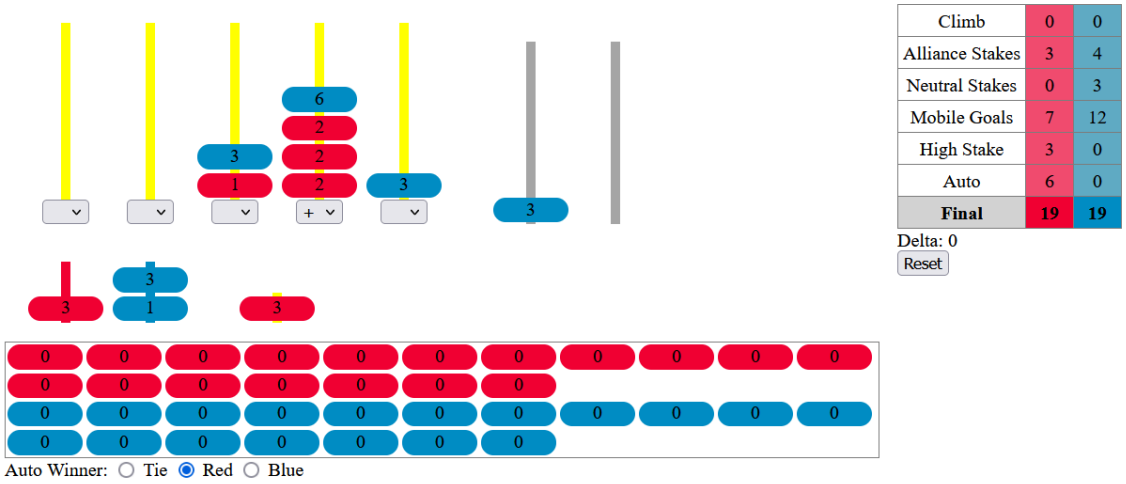
Continued from **Google Sheets Scoring Calculator** (Pg. 10)

As an alternative to the Google Sheets scoring calculator, we set up a GitHub Pages website to create a new scoring calculator. This allows us to create the calculator using HTML, CSS, and JavaScript, which means we can customize the calculator much more than Google Sheets, which only allows for basic formulas. Since the website is hosted on GitHub, anyone can access it, meaning that once it is finished, we could send it out to other teams as an alternative to the V5RC Hub scoring calculator.

To improve user experience, we included a full score breakdown on the right side, along with a Delta to show the score difference between the two teams. Additionally, each ring will display its score value for full transparency of where points come from.

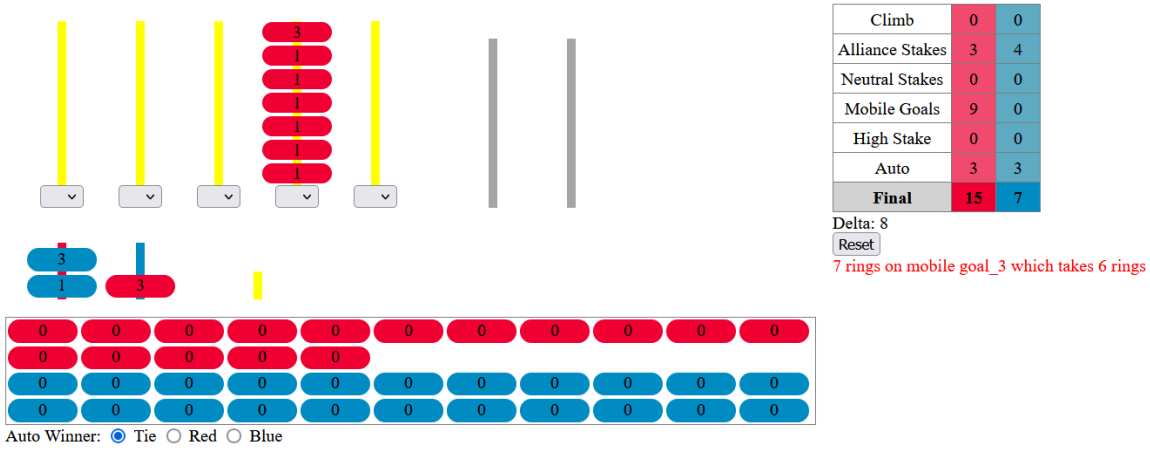


GitHub Pages Website Scoring Calculator Layout



Example Usage of Scoring Calculator Website

Since it is still in its early stages, there are some illegal actions that a user can do. For example, more than the maximum number of rings per stake can be scored, and the wrong colored ring can be scored for points on an alliance stake. Additionally, all 5 mobile stakes can be placed in a +/- corner, which should have a maximum of 2 per modifier. Currently, an error message appears telling the user that their action was illegal.



Wrong Colored Rings on Alliance Stakes and 7 rings on Mobile Goal

Future Plans

Going forward, illegal actions will not be able to be performed instead of having an error message pop up. Additionally, climb scores still need to be added, as well as a skills section.

Continued in **Correct Match Scoring Calculator** (Pg. 80)

Notebook Support Software

Focus: **Notebook Support Software**

By: **Richie Sommers** Date: **06/10/2024**

With the selection of Obsidian as the notebooking platform of choice, we started to look into ways to streamline and customize it to our specific needs.

Standardization

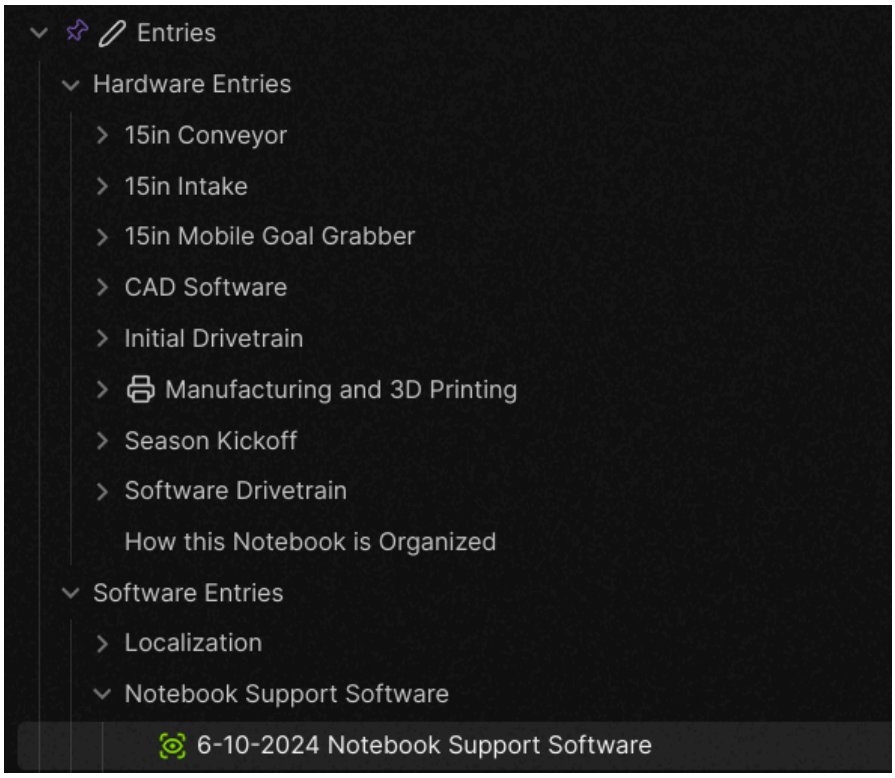
The team notebook is a collaborative project with many contributors. Team members of various engineering disciplines, majors, and styles work together to create a document that contains each persons expertise and a more informative product than a single person could write on their own. However, there are as many styles as there are authors which can lead to idiosyncrasies that detract from the standard, easy to follow format. Some of these issues, such as tense and general grammar issues, could only be written up as guidelines to follow and enforced by proofreading entries. However, some guidelines could be enforced without human nuance.

Specifically, the following issues were observed as options for programmatic standardization

- Acronyms and Brand Names: VEXU, VEX U, Vex U, etc.
- Metadata
    - Date: Each entry requires a date. If that date is missing or mistyped, the final product will not be in the correct order.
    - Author: Each entry requires one or more authors.
    - Process Step: The notebook is organized according to the engineering design process. With a few notable exceptions, each entry should be marked as part of that process so we can organize the notebook accordingly

Notebook Generator

Obsidian provides a great way to organize information for digital creation and access. We organize information into subteam and project. Compared to a single large document, it is much easier to find, work on, or learn from an entry.



Organization of Notebook Entries in the Obsidian Editor

However, for the print or PDF form of the notebook, a tool to join together and serialize these entries is needed.

Notebook Backup

Through RIT, the team has a shared Google Drive with which to host the files that make up the notebook. Obsidian operates on a standard file system interface with no hidden data stored by any third party. So, by utilizing the Google Drive desktop app (or rclone for linux-using members), members can all get access to the notebook.

Unfortunately, while Google Drive provides a hosting solution and some limited backup capabilities for text files, the backup abilities are not very easy to use. Also, it would be impractically difficult to apply permissions to each individual file or folder in order to prevent accidental or purposeful vandalism. So, a more complete backup system is worth pursuing.

Continued in **Entry Checker** (Pg. 15)

Entry Checker

Focus: **Notebook Support Software**

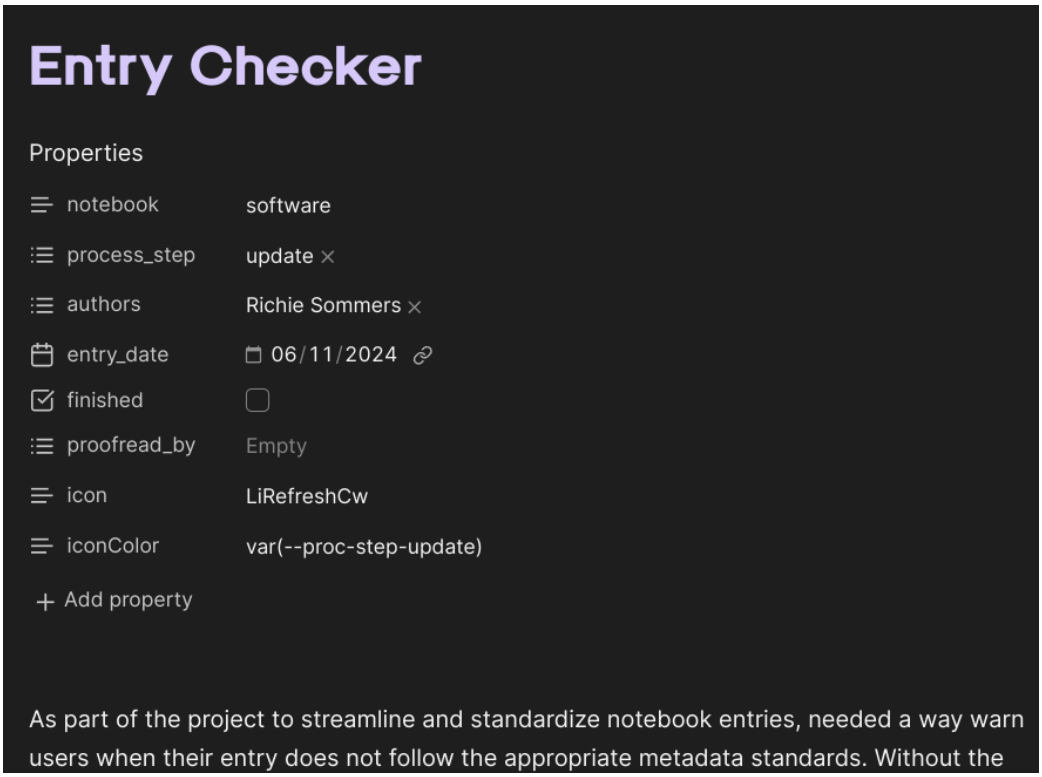
By: **Richie Sommers**

Date: **06/11/2024**

Continued from **Notebook Support Software** (Pg. 13)

As part of the project to streamline and standardize notebook entries, we needed a way warn users when their entry does not follow the appropriate metadata standards. Without the appropriate metadata, entries can not be sorted, searched, or joined into the final notebook correctly so it is very important to get this right.

Additionally, to take a more proactive approach, we created templates that members can use to quickly fill in the information that does not change often. Obsidian offers shortcuts to insert a template into a new note which allows a friction-free way to get 90% of the way to metadata completion.



The Metadata for this Note

Dataview

There is a widely used, community driven plugin called Dataview for Obsidian. This plugin allows file contents to be queried using a language similar to SQL (a language traditionally used for managing databases).

```dataview
TABLE WITHOUT ID authors as "Authors", round(sum(map(rows, (r) =>
default(r.file.size, 0)))/6)  as "Words Approx.", sum(map(rows, (r) =>
default(r.file.size, 0))) as "Size (bytes)"
FROM "Entries"
WHERE process_step != null
FLATTEN authors
GROUP By authors
SORT sum(map(rows, (r) => default(r.file.size, 0)))
```

Dataview Query in Edit Mode

Authors 10	Words Approx.	Size (Bytes)
Colt Franklin	118	707
Samuel Radulski	180	1079
Tyler Kennedy	530	3178
Aidan Kelley	740	4439
Victor Rabinovich	747	4481
Jack Cammarata	1957	11741
Mae Subramanian	3062	18371
Ellie Bancroft	6049	36295
Zoe Rizzo	6642	39851
Richie Sommers	7323	43940

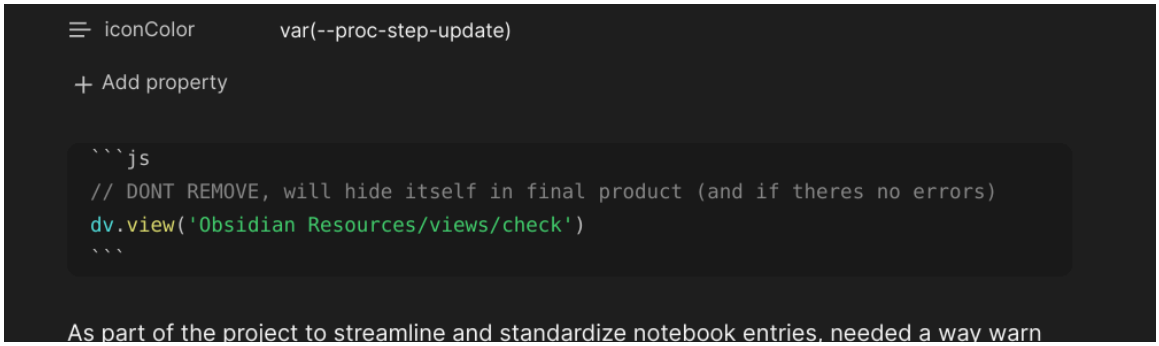
Dataview Query in Preview Mode

This was originally considered for use for this project. However, it has a restricted set of features that could not do all that we wanted it to do.

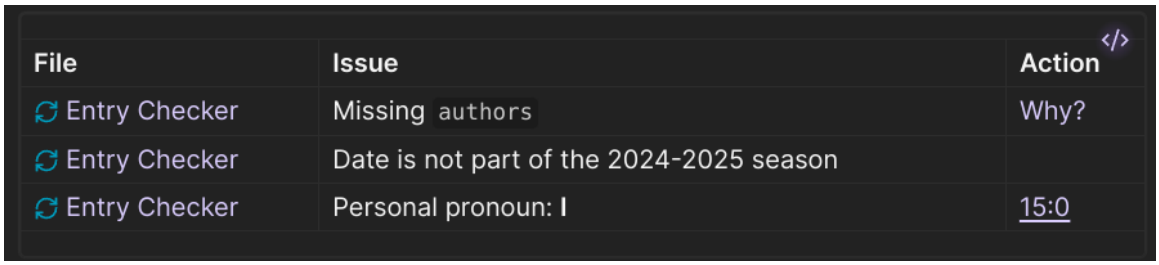
Dataview JavaScript View

Luckily, Dataview offers a JavaScript API to access the same underlying data. This JavaScript API can be combined with custom CSS to output arbitrary, custom elements into the document. When viewed within Obsidian, these snippets of code will be executed and shown to the user. This combination of JavaScript and CSS when used in this way comprise a 'view' in Dataview parlance.

Using this API, we created a view that we include in all entries that checks on metadata and file contents.



The underlying syntax to activate the `check` view



The `check` view when viewing the file normally

The current implementation of the `check` view warns the user about the following:

- Missing Metadata
  - `authors`
  - a `date` outside of this season
  - a missing or invalid `process_step` (the parameter we use to track the type of entry)
- Content Issues
  - Use of personal pronouns other than we, our, and us. To maintain professionalism and give a greater focus to ***what*** is occurring rather than ***who*** is doing it, we aim to avoid personal pronouns. However, from past years we found that disallowing the use of we, our, and us made for very clunky and repetitive sentence structures. After some discussion, we decided that we could allow these as they lead to more natural sounding entries without making entries overly personal or possessive.

As we develop our notebooking format and procedure over the season and coming seasons, this check may have to adapt. Luckily, its source is included in the same project as all of the notebook entries. So, it will be preserved and tracked as it develops.

Continued in **Internal Notebook Generator** (Pg. 22)

Vex Debug Board

Focus: **V5 Debug Board**

By: **Richie Sommers**

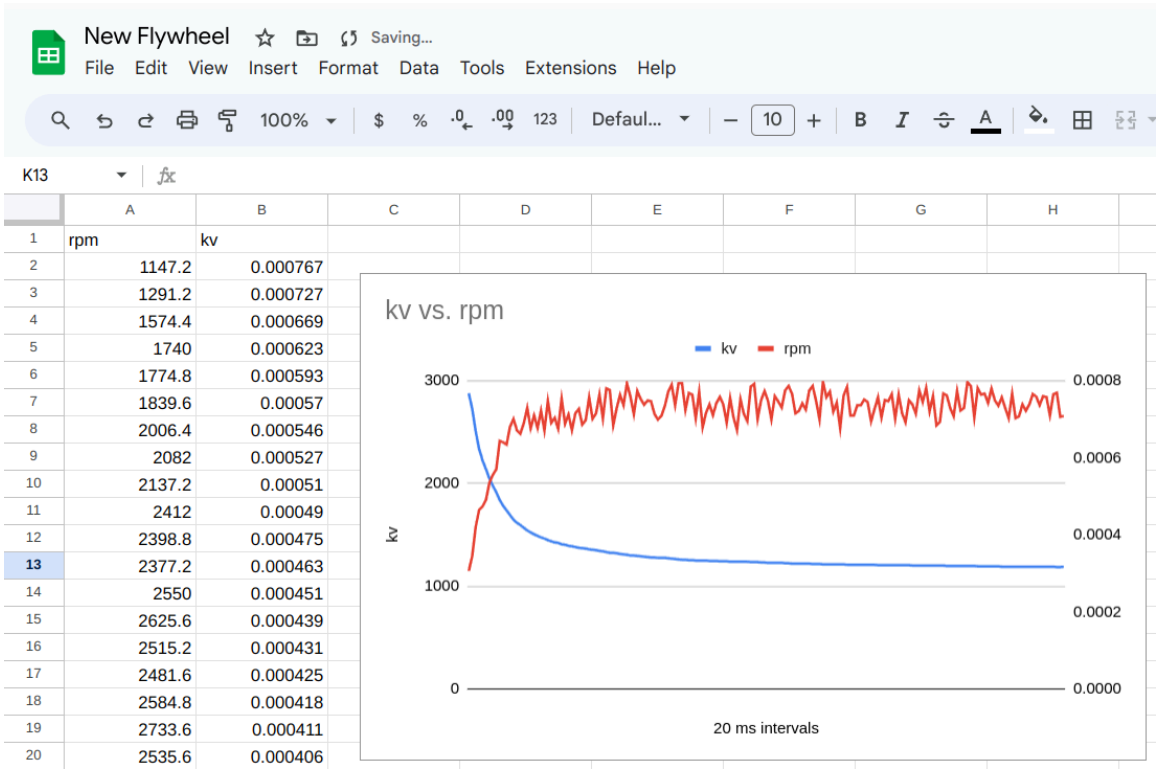
Date: **07/01/2024**

Missing Documentation

The electrical and hardware design of the Vex Debug Board was done in a previous year, by a member no longer with us, before the team emphasized formal and complete documentation. As such, some information about original design decisions does not appear in this notebook.

Developing robot control systems requires accurate data of what the robot experiences. In the past, we would use one of two options to get a view into what the robot was seeing.

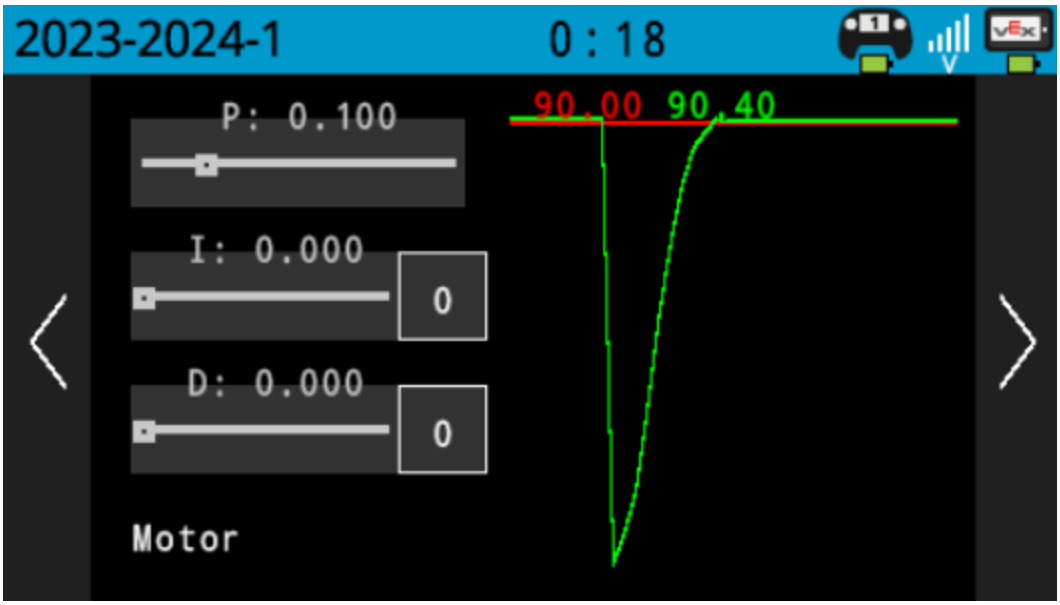
Originally, the best path we could take was to print out values in a comma separated value format from the brain, through the vex controller into our IDE. Then we would import this data into a spreadsheet and graph or otherwise analyze it. This solution was slow, clunky, and often failed if we tried to print too much data. The serial link would randomly drop characters which could render a line completely useless meaning the programmer would have to edit out lines manually further adding to the slow time from data to analysis.



	A	B
1	rpm	kv
2	1147.2	0.000767
3	1291.2	0.000727
4	1574.4	0.000669
5	1740	0.000623
6	1774.8	0.000593
7	1839.6	0.00057
8	2006.4	0.000546
9	2082	0.000527
10	2137.2	0.00051
11	2412	0.00049
12	2398.8	0.000475
13	2377.2	0.000463
14	2550	0.000451
15	2625.6	0.000439
16	2515.2	0.000431
17	2481.6	0.000425
18	2584.8	0.000418
19	2733.6	0.000411
20	2535.6	0.000406

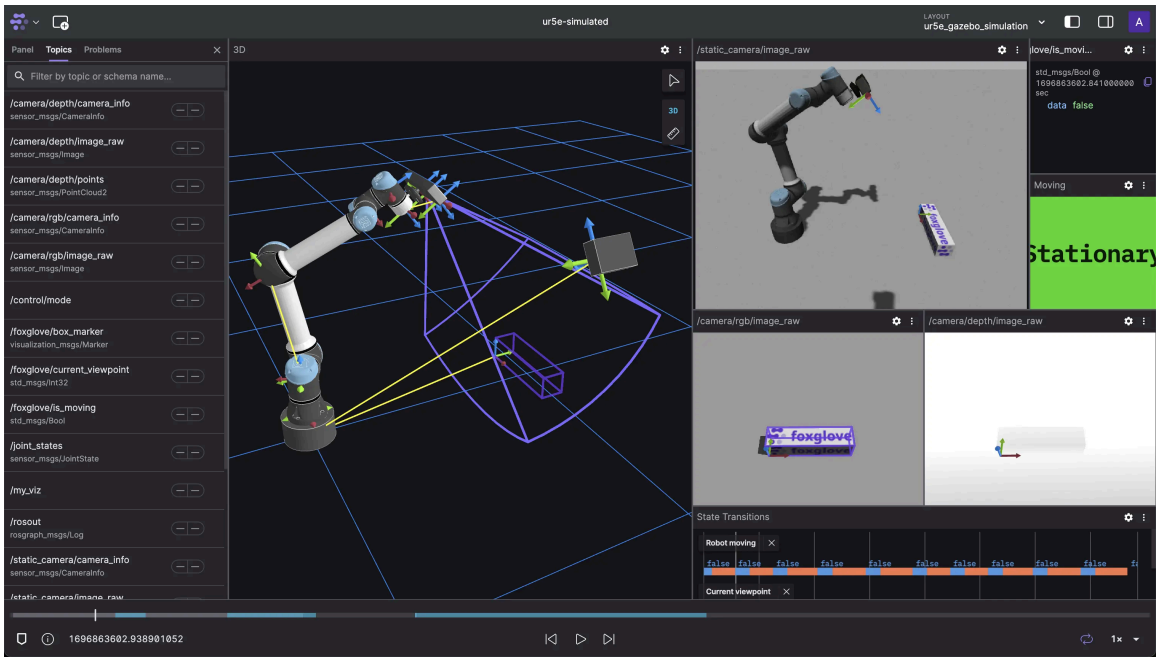
Tuning a flywheel control system using printf, copy-paste and Google Sheets

To remedy this unpleasant situation, we developed a graphing widget for the brain screen. This allowed a faster, qualitative view into the system - we could easily see an oscillation in the control system as it happened. However, it offered nothing in terms of raw numbers and there was no quantitative data past what we could make out. Additionally, when tuning a drive train, the robot is moving and members would have to follow it around trying to see how it performed. While providing a better feedback time, this was not an all around great solution.



Tuning a PID controller on the brain screen

As a true fix for this issue, we developed an idea for a "Vex Debug Board" (VDB) that would allow the brain to transmit information wirelessly to a computer where the data could be analyzed in real time or saved for later analysis. Live telemetry achieved this way could then be fed to a visualization platform such as RViz from the ROS project or Foxglove Studio, a standalone application for visualizing robotics systems.

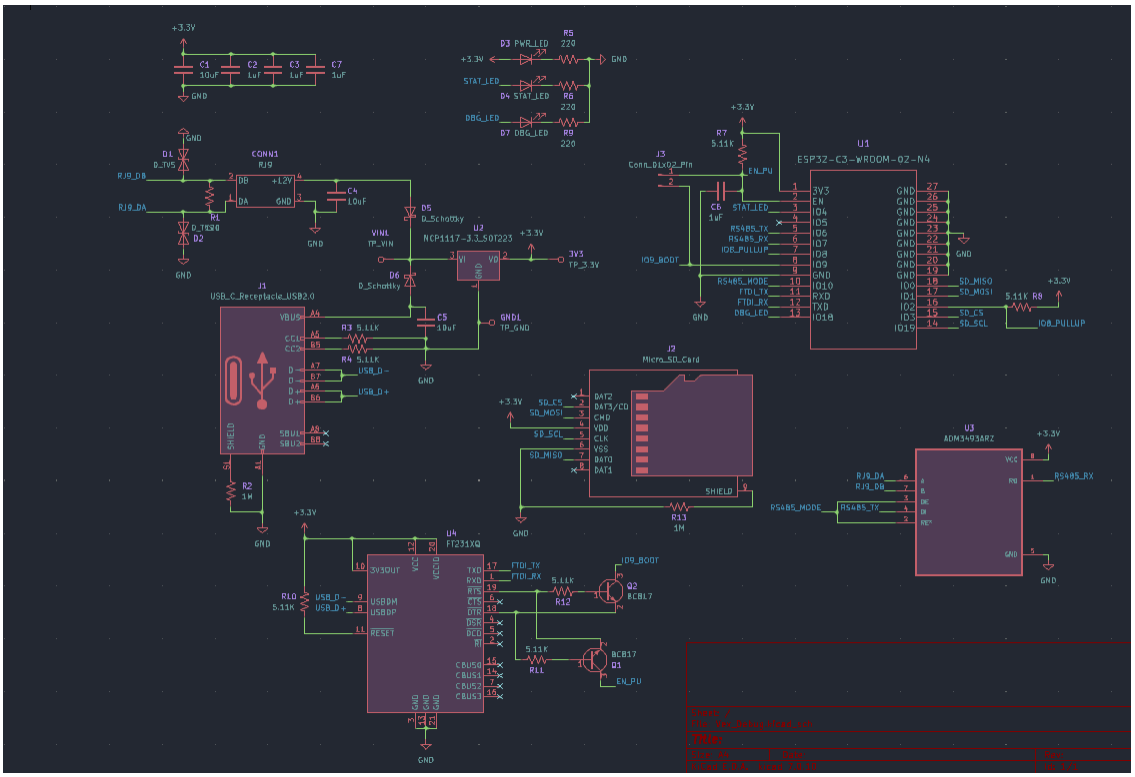


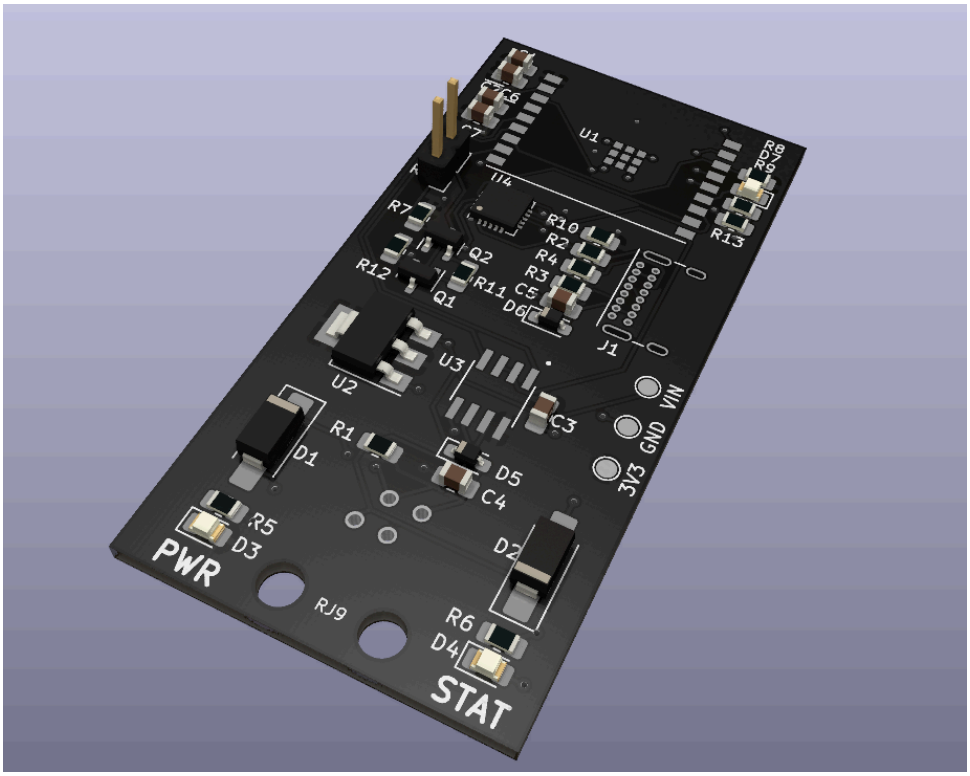
An example of Foxglove used for developing a robotic arm

With fast, live, error-free telemetry, quantitative measurements of the robot could be made and control algorithms could be more quickly tuned. While we could do these before, the generational leap in speed that the VDB will provide will dramatically lessen iteration time meaning better algorithms can be developed in less time.

Hardware

The Vex Debug Board is an ESP32 based system. The ESP32's built in support for WiFi and Bluetooth simplifies the circuit significantly compared to an external radio. Alongside this microcontroller, there s a USB-C port for programming, a Vex "Smart Port", status LEDs, and an SD card slot. The board can be powered either from the USB port or through the "Smart Port" simplifying integration into a robot.

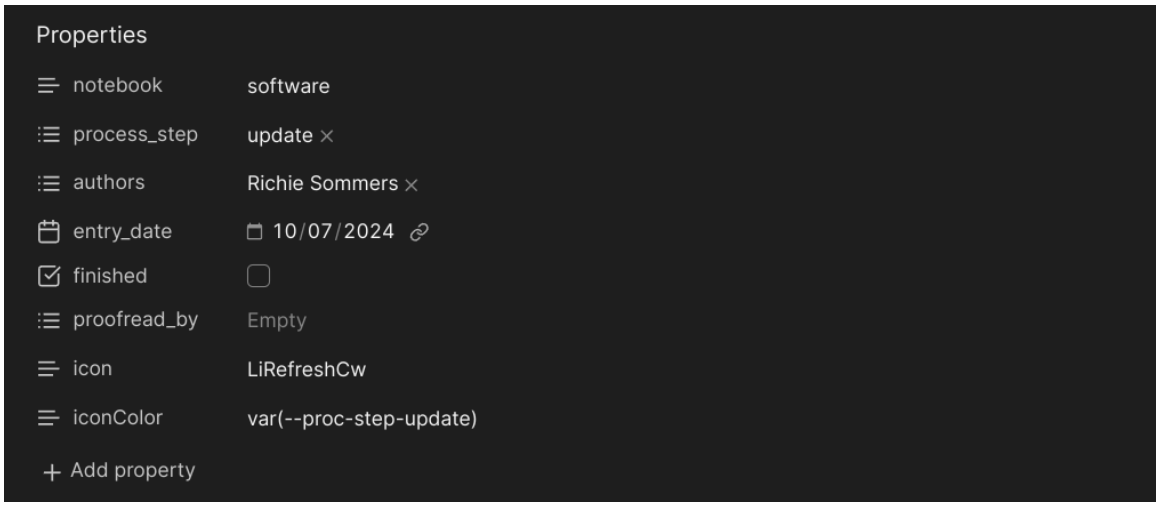




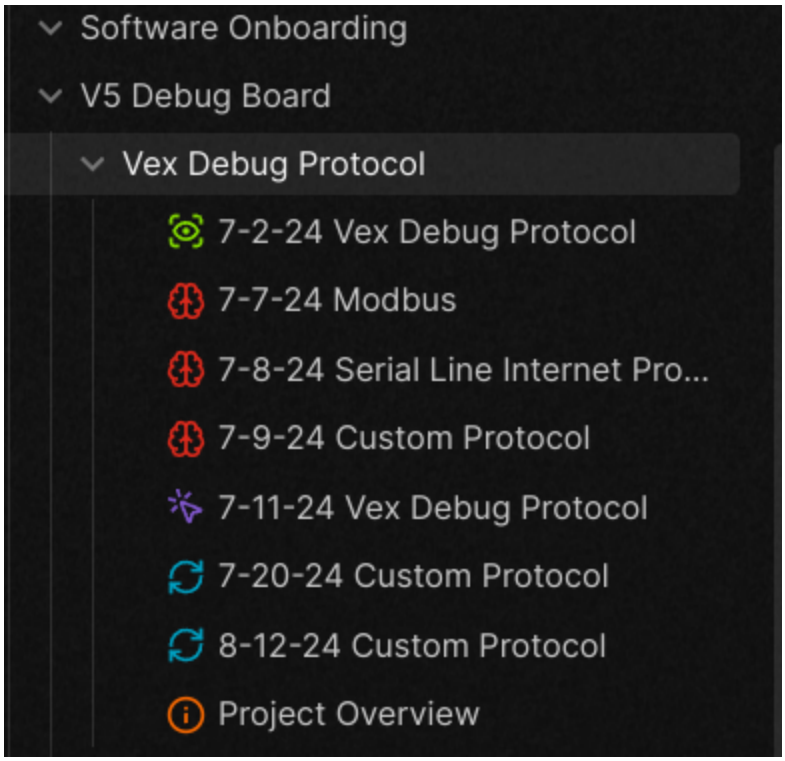
Continued in **Initial Web Server** (Pg. 31)

Continued from **Entry Checker** (Pg. 15)

Given the directory structure we use to organize the notebook, we need to transform the tree structure of Markdown files into a printable PDF. Using metadata such as a notes position in the file tree, YAML frontmatter metadata, and content, we must serialize the entire years worth of work into a single stream of information.



Metadata for an entry. Some metadata such as date, authors, and notebook are used to organize the notebook. Others are used to organize the team or visually distinguish entries in the editor.



Directory structure with icons and colors provided by metadata

At this point, it has not been decided whether to serialize by date (entries are in the order they were written) or by project (entries are organized alphabetically by project and within each project, sorted by date). Both approaches have their benefits and we are still undecided which to aim for. So, we aim to not finalize anything based on assumptions of one or the other.

Dataview JavaScript Implementation

This implementation of the notebook generator uses the concept of a Dataview 'view' (as discussed in the 'Entry Checker' write up.

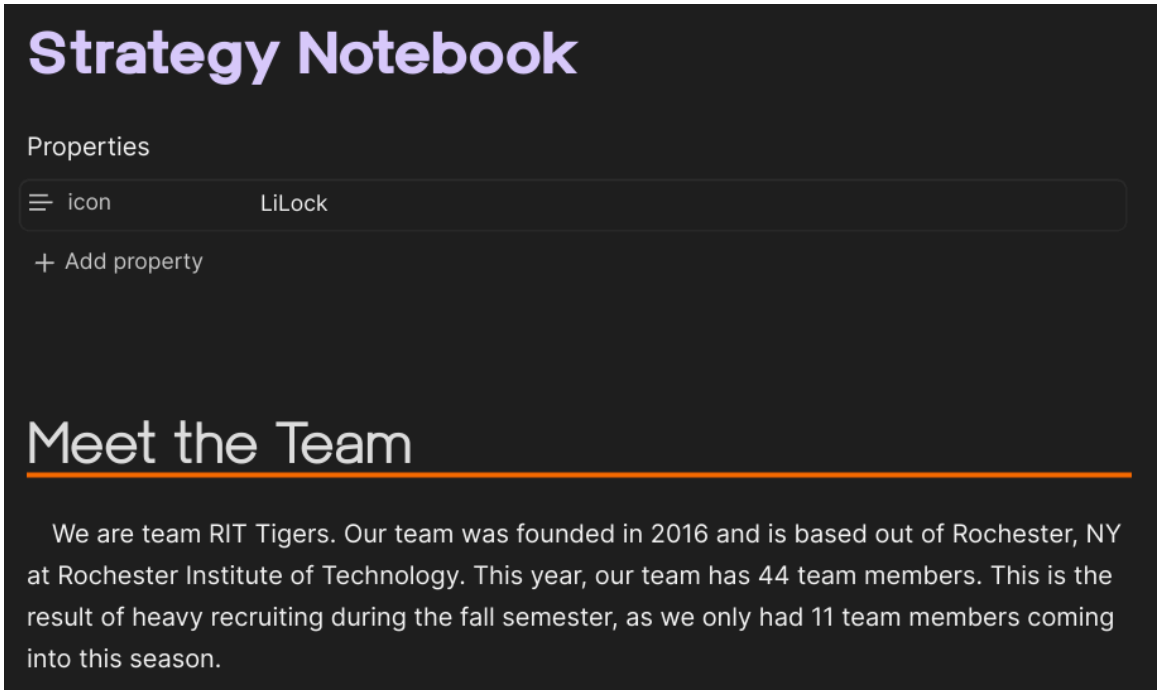
The current implementation uses the Dataview API to find all entries from our `Entries/` folder. It then filters this list by whether or not it has any metadata errors using the same logic as the Entry Checker project. Then, we filter the entries into two lists, frontmatter and regular entries. Frontmatter entries contain information needed to understand the rest of the notebook as well entries that are not engineering design process focused such as the entry for introducing the team members. These entries are set aside and only brought in if requested by the arguments to the view.

```js
// Ordered list of all special (non eng design process entries)
const frontmatter = [
    "Strategy Entries/How This Notebook Is Organized",
    "Meet the Team",
    "Meet the Bears Behind the Bots",
    "The Engineering Design Process"
];

await dv.view('Obsidian Resources/views/notebook', {frontmatter: frontmatter,
notebook: 'strategy'});

```

Finally, the remaining entries are sorted by date and rendered using html primitives and the built-in markdown renderer. When the user moves their cursor out of the codeblock that defines the notebook, it is rendered thusly.



To preview individual projects, we adapted this code for a 'Project Overview' view such that for a given project, we can get a preview of what it will look like in the final output. The Project Overview is a template that writers can insert into the folder with their entries. It will then check the folder, find all the entries in it, and concatenate them in date order like the final notebook. However, it only takes the entries for a specific project which allows an author or learner to see the entire progression of the project in one place.

Continued in **External Notebook Generator Update 1** (Pg. 50)

Vex Debug Protocol

Focus: **Vex Debug Protocol**

By: **Richie Sommers**

Date: **07/07/2024**

The communications from the Vex Debug Board (VDB) to the users computer is determined by IEEE internet standards and the visualization software we choose to use. However, the protocol used to communicate from the VDB to the brain over the RS-485 connection is not forced upon us.

This protocol must accomplish the following:

- Have minimal overhead - the serial port is not the fastest and every byte saved counts
- Runtime adaptability - It would be a hassle have to re-flash the VDB every time you want new data. The protocol should be able to handle changing data requirements when the robot programmer decides to send different data.
- Ease of use - the protocol should be easy enough to add and remove from robot code. Robot programmers should not have to know the ins and outs of the protocol in order to get benefits from using this tool.
- Bidirectional communication - Not only would we like telemetry from the robot, sending tuning parameters without having to rebuild and redeploy the project saves valuable time.

Continued in **Modbus** (Pg. 26)

Modbus

Focus: **Vex Debug Protocol**

By: **Richie Sommers**

Date: **07/08/2024**

Continued from **Vex Debug Protocol** (Pg. 25)

The Modbus communications protocol is commonly used for programmable logic controllers (PLCs) in industrial settings. It can be implemented on an RS-485 connection, the same that the Vex Brain has, as well as over a TCP/IP. The Modbus protocol is driven by a controller that can initiate a transaction to many peripherals who can then respond with data or with an error when necessary. The controller requests to execute a specific *function*, at a certain *address*, with a variable amount of extra data needed for the request.

There are two forms of the Modbus protocol: RTU or ASCII. In our scenario, we can send arbitrary binary data so we would most likely use the RTU version.

start	address	function	data	crc	end
at least 3.5 chars	8 its	8 bits	n * 8 bits	16 bits	at least 3.5 chars
Modbus RTU Frame Format					

While this is a fairly simple, recognized standard, this protocol presents some issues.

- It only supports 16 bit and 1 bit data by default. While other data types can be added on top, it does add complexity.
- It the one controller multiple peripherals does not lend itself to easy bidirectional communication. A polling approach could be taken to check for updates on either side but a symmetric protocol makes more sense.
- The biggest of all - it is not clear how to make the protocol easily adaptable to changing data types and data channels. Since we would have to do a fair amount of custom work on top, it may be easier to just define a fully custom protocol.

Continued in **Serial Line Internet Protocol** (Pg. 27)

Serial Line Internet Protocol

Focus: **Vex Debug Protocol**

By: **Richie Sommers**

Date: **07/08/2024**

Continued from **Modbus** (Pg. 26)

Serial Line Internet Protocol (SLIP) is a protocol that defines a standard for a full Internet Protocol over a serial connection. This effectively would allow the brain itself to access the internet and devices connected to the ESP32's network. In our case, for example, we could operate the foxglove websocket protocol from the brain, using the Vex Debug Board as nothing more than a switch. A full TCP/IP stack would give pretty much unlimited flexibility in terms of what we want to accomplish without ever having to update the code on the ESP32. So long as it correctly routed packets between its serial connection and its wireless radio, it would never need changing. This flexibility was quite enticing to certain programmers who saw exciting and fun (though not necessarily beneficial to points scored in a season) uses for this technology.

However, this approach comes with major downsides:

- If the debug board simply acted as a network switch, we would require a suite of IP tools on the brain. A full internet stack on the brain would be quite a complex piece of software somewhat unrelated to robotics. Our code base would balloon in size without all that much benefit
- While the Internet Protocol provides incredible flexibility, there is overhead needed to make the internet work that the robot does not need and it could incur substantial performance penalties as the robot struggles to keep up with communicating rather than doing robot things.

Continued in **Custom Protocol** (Pg. 28)

Custom Protocol

Focus: **Vex Debug Protocol**

By: **Richie Sommers**

Date: **07/09/2024**

Continued from **Serial Line Internet Protocol** (Pg. 27)

A custom protocol offers maximal flexibility in terms of what we want the robot to communicate to the Vex Debug Board(VDB). Importantly, it allows us to strip out all strictly unnecessary overhead sending only absolutely necessary bytes to make maximum use of our limited bandwidth. The incredible domain specificity of this protocol allows us to make many choices that would simply be invalid for a more general use protocol.

The broad outline of our plan is a bidirectional, two phase, binary protocol. Both the brain and the VDB can send and receive information to each other - only constrained by what shapes of data can be constructed by build in primitives. The first phase consists of the Vex Debug Board and Brain finding each other and advertising **channels**. **Channels** are a source of statically typed data and both sides of the communication maintain their own list of channels they can receive and channels they will send. The advertising of a channel consists of each party sending an **Advertise** message containing a list of **schemas**. These **schemas** contain the human readable names for data within the channel as well as containing information about how to decode the forthcoming **Data** messages.

The second phase takes up most of the time that the robot is running wherein both parties can send **Data** messages consisting of raw binary packets. These packets are not self-describing so the advertising of channels must have happened before **Data** packets can be sent. The lack of self description does make the protocol slightly more complicated as compared to other protocols where the data needed to decode a message is contained inside the message meaning no advertising is needed. However, the two phase approach allows for less bandwidth used as we do not need to re-transmit the information needed to decode data, just the data itself meaning faster update rates or larger data packets on the same amount of bandwidth.

At this point, though not every byte of the protocol has been decided on, some prototypes have been constructed showing that this is possible and provides an API that is fairly easy to use. These prototypes do not yet have the "header" bytes that tells what channel number this is for example, nor do they contain any framing mechanism for splitting apart packets. These features will come later.

The biggest con of this method is that we must handle all layers of protocol ourselves. That is, from the physical layer up we must implement all the code needed to frame packets, detect errors, and decode information.

Schema Packet Prototype

```
Schema Packet:

00 4d 6f 74 6f 72 20 31 00 05 00 00 00 03 50 6f
73 69 74 69 6f 6e 28 64 65 67 29 00 03 76 65 6c
6f 63 69 74 79 28 64 70 73 29 00 04 54 65 6d 70
65 72 61 74 75 72 65 28 43 29 00 03 56 6f 6c 74
61 67 65 28 56 29 00 03 43 75 72 72 65 6e 74 28
25 29 00
Advertise Message Size (bytes): 83

Schema (Human readable interpretation of above data):
Motor 1: record[5]{
  Position(deg):        float
  velocity(dps):        float
  Temperature(C):       uint8
  Voltage(V):   float
  Current(%):   float
}
```

Data Packet Prototype

```
Data Packet:
9a 99 89 42 33 33 11 43 23 81 95 f3 3f 48 e1 2a 41

Data Message Size (bytes): 17

Data (Decoded with information from schema):
Motor 1: record[5]{
  Position(deg):    0.8
  velocity(dps):    86.4
  Temperature(C):   35
  Voltage(V):       1.645
  Current(%):       5.96
}
```

Continued in **Vex Debug Protocol** (Pg. 30)

Vex Debug Protocol

By: **Richie Sommers**

Date: **07/10/2024**

Continued from **Custom Protocol** (Pg. 28)

After considering the options, we have decided to implement a custom protocol. Though we can not use any off the shelf libraries for it, the flexibility and power it gives us makes it worth the upfront development cost. We will be able to optimize to our hearts content without having to delve into 3rd party libraries and strip out parts which would remove the benefit of using a 3rd party to save time.

	Modbus	**SLIP**	**Custom**
Ease of use	+	-	+
Runtime Adaptability	-	+	+
Low Overhead	+	-	+

Continued in **Custom Protocol** (Pg. 33)

Initial Web Server

Focus: **V5 Debug Board**

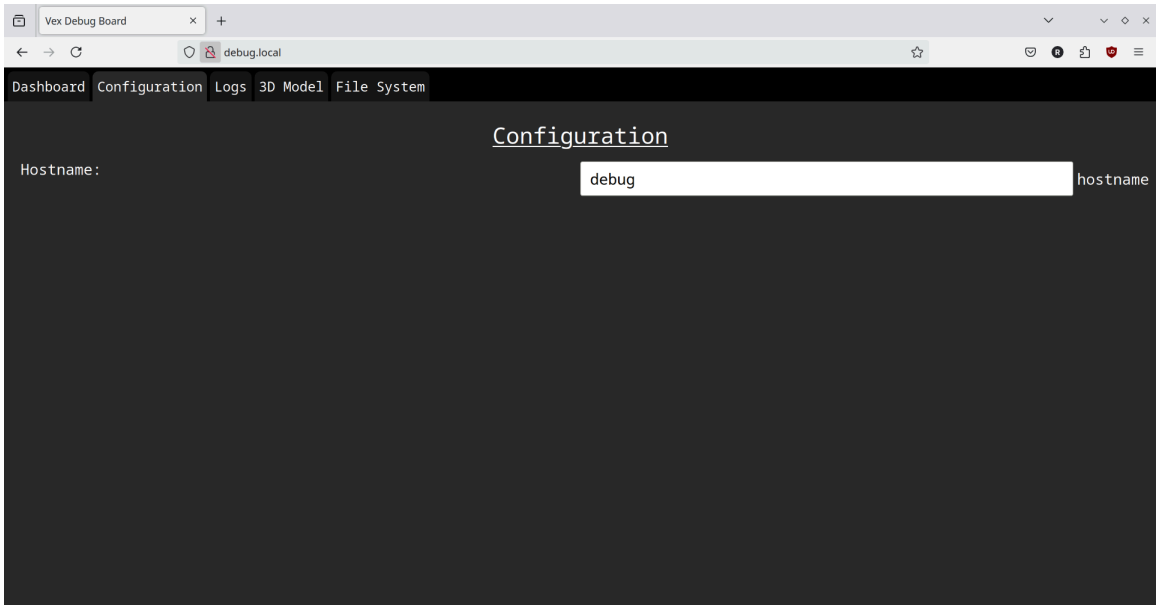
By: **Richie Sommers**

Date: **07/13/2024**

Continued from **Vex Debug Board** (Pg. 18)

We desired a way to allow easy configuration of the Vex Debug Board(VDB) without needing to rebuild and redeploy software. This not only takes a fair amount of time, but is also unavailable to those without the entire `esp-idf` tool chain installed.

A web interface was the obvious solution to this. A simple web app could allow users to connect and reconfigure wirelessly without any external tools. This webserver could also serve other purposes such as checking the status of the board or checking logs. Eventually, we would like to support a filesystem on the SD card in order to hold 3D models of the robot to use in Foxglove or other visualization platforms. This web portal could be used to upload and preview this model.



An early, in development look at the web interface

We chose Elm for the front end language as it was a preferred language for front end development by the writer of the web app and the strong guarantees it makes. Elm is a pure functional programming language designed for robust and performant web interfaces that promises zero runtime exceptions. That is, if it compiles, no front end crashes are possible (hardware issues not withstanding). Additionally, it provides a powerful "time travelling" debugger that can be easily embedded into the web page meaning we can debug the website (and even the back end responses) easily. This is vital as debugging the server is considerably more difficult because it runs on a micro controller compared to debugging on a traditional server running on a 'standard' server computer.

Elm compiles down to HTML and Javascript using its own compiler. From this (or any other front end code) we needed a way to embed the front end such that the ESP32 can find and serve it. To accomplish this, a custom command was added to the esp-idf cmake project that would first make the elm project, then minify the code, and finally gzip the resulting output. This produced an output small enough to be flashed into the ESP32s memory alongside all of the C code that runs the board. This means no SD card will be needed by default and the board will be perfectly usable without one.

The overall design of the front end is a simple web app with a JSON REST backend. After the initial page load, no UI data is requested from the ESP32. In fact, in a production environment, we could tell the browser to cache the entire UI which would reduce the time to load without any ill effects as all "live" data is brought in through HTTP requests to the board. This also provides a more pleasant developer experience as, when working on the front end, a programmer can run the UI on their computer while still sending backend requests to the board's REST server meaning the time for an edit-compile-run is limited only by the Elm compiler (almost instant) rather than the time it takes to deploy to the board (upwards of 30 seconds).

Custom Protocol

Focus: **Vex Debug Protocol**

By: **Richie Sommers**

Date: **07/19/2024**

Continued from **Vex Debug Protocol** (Pg. 30)

This week, effort was put towards finalizing the custom protocol used to communicated between devices. There are two "layers" of the protocol - we have chosen to call these the application layer and the link layer. The link layer describes the way that data is transferred over the physical medium of the serial cable and the application layer describes the information the protocol uses to exchange data. The current protocol is described here.

Terms

Here are brief definitions for the terms used later on. More detailed descriptions are available in the *Application Layer* and *Link Layer* sections

Checksum - For this protocol, we use a CRC32 checksum to check the integrity of the packets. We calculate the checksum and place it at the end of the packet. Then, upon reading, we recalculate the checksum from the data we received and compare it with the encoded checksum. If they match, we can continue to decode the data and use it. If they do not match, we ignore the packet.

Channel - A channel is a logical unit of data. It is defined by a **Schema** and is assigned a **Channel ID** to identify it. Data transmitted under a channel's ID must fit the **Schema** in order to be correctly decoded.

Channel ID - an integer uniquely identifying a channel

Delimiter - A zero byte used to mark the boundaries between packets for bytes transferred on the serial link

Header - For this protocol, a header describes the type and function of a packet. More info about the structure of a header is shown below

Packet - A certain number of bytes that are logically grouped together, beginning with a **Header**, containing byte data, and terminated by a **Checksum**

Wire Packet - A Consistent Overhead Byte Stuffing (COBS) encoded, zero delimited string of bytes that is ready to be sent over the serial link. It is distinct from a Packet in that the decoding of robot data does not really concern itself with this level - It is an implementation detail. If we wanted to send **Packets** over a different system, the Wire Packet could be encoded differently and so long as the **Wire Packet** Encoder/Decoder functioned consistently. the protocol would not care.

Application Layer

The application layer performs two functions - broadcast/discovery of data channels and sending/receiving of data. Messages in the application layer are described by packets.

Packet

An application packet can have the following data.

Header - a single byte of information identifying the type and purpose of this packet

Channel ID - a single byte describing what channel of data this message pertains to. The value of 0 is reserved for protocol related activities leaving 255 channels available for user data.

Data - This section is optional as the **header** and **channel id** may describe the entire message. When used, this section can encode a **Schema** or **Data**

Checksum - a CRC32 checksum used to verify the integrity of a packet

At minimum, the packet must contain a **Header**, **Channel ID**, and **Checksum** meaning the minimum packet size is 6 bytes

header	channel Id	data	checksum
1 byte	1 byte	X	4 bytes

Packet Structure. Note: Data can be an arbitrary length but the last 4 bytes must always be the checksum

Header

The header is a bitfield containing the following information.

Packet Type: Broadcast or Data

Packet Function: Send or Acknowledge

Both data and broadcast packets can be sent but only Channel broadcast messages will be acknowledged. The receiver needs the description of the channel to properly decode the information so the sender uses **Acknowledges** (or the lack thereof) to know if the receiver is ready for data. If no acknowledge is received after a given timeout, the broadcast is re-sent. The sender does not expect or require an acknowledge for data packets - we want data to flow as fast as possible and the overhead of acknowledging could lead to slowdown that would better be spent sending more up to date information.

At the time of writing, the remaining 6bits are unused and left open for future needs

Channel Broadcast/Discovery

A channel broadcast packet consists of a header, the channel id of the channel its describing, the schema definition for that channel, and a checksum.

The schema definition provides the structure and names of different fields in the following data packet. This gives the receiver the information it needs to reconstruct structured data from the flat buffer of values it will receive in **Data packets**.

Channels can be updated as time goes on simply by rebroadcasting the channel. This is necessary as the ESP32 side will be powered on while code is continuously redeployed to the robot and the channels it sends change.

Schema

A schema type can be one of the following: `Record`, `String`, `Double`, `Float`, `Uint8`, `Uint16`, `Uint32`, `Uint64`, `Int8`, `Int16`, `Int32`, or `Int64`.

A schema contains a 8-bit type byte, a 0 terminated string name, then any internal data needed to represent the type.

Unless otherwise specified, these types correspond with the standard C types. All integers are encoded as little endian and all numeric types are decoded by a simple copy from uint8 memory into the underlying type.

A record is very similar to a C struct consisting of named fields. These fields can be of any type. Its schema is described like all others (A type byte and a name) but then contains a number representing the number of fields. Then, each field is encoded as if they were the schema type. When decoding a schema, the decoder will see that this element is a `Record`, find the number of fields, then read in that many more fields into the record.

An example of a schema. A record with 5 fields named 'Motor 1. Then, 5 fields of varying names and types.

As of now we have not implemented them, but arrays and enumerations could prove helpful. If we find a need we will discuss their implementation at a later date

Channel Data

A data packet consists of the header, the Channel ID for the channel its data corresponds to, a buffer of data, and a checksum. The encoding simply walks the tree of a schema writing values of the first children first in the case of records. For example, the Motor schema shown above would produce a data packet of the following form

```
header
channel id
4 byte float for Position(deg)
4 byte float for Velocity(dps)
1 byte uint8 for Temperature(C)
4 byte float for Voltage(V)
4 byte float for Current(%)
checksum
```

This packet will be 23 bytes long

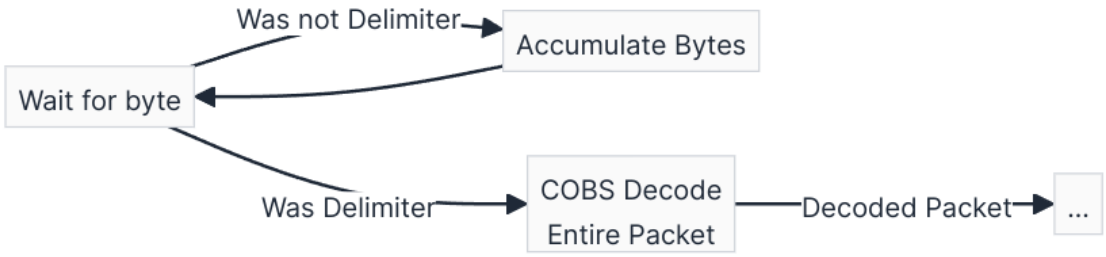
Testing the protocol with a "loopback" cable connected from one smart port to another

Link Layer

The link layer provides the communication medium for the application layer. it is responsible for transmitting packets over some physical medium in a way that the other device in the exchange can reconstruct individual packets. This is implemented using Consistent Overhead Byte Stuffing (COBS) with a delimiter of 0. The encoding phase reinterprets the passed in data such that no 0s appear. Then, a zero is pre-pended and appended to mark packet boundaries. Then, the packet is sent over the serial link.

On the receiving end, the 0s mark the start and end of packet. From here, the receiver can decode to get the data 0s back, before passing the packet to the **application layer** for using.

Reading from the wire



Writing to the wire

Implementation Notes

Both the VEX Brain and the ESP32 support C++ but have very different serial IO APIs. We can reuse much of the protocol code so long as we separate the IO from the rest. To do this, we implemented the protocol in two main classes. The first, the *Registry* class, handles everything in the application layer - Parsing packets, writing packets, handling Acknowledge commands, broadcasting, and sending data. This, as well as all of the type definitions and helper functions, can be used verbatim on both the ESP32 and the VEX Brain. The second class, the *Device* class handles all IO and encoding of Application packets into COBS packets. This allows us to reuse all of the code except for the *Device* class.

Continued in **Custom Protocol** (Pg. 38)

Custom Protocol

Focus: **Vex Debug Protocol**

By: **Richie Sommers**

Date: **08/12/2024**

Continued from **Custom Protocol** (Pg. 33)

Since the last entry, one major update has occurred. This change revolves around the "flow control" of the protocol. To avoid the VDB and Brain talking over each other on a half duplex connection, we have decided to make the protocol controlled by one side acting as a controller and the other responding to the controller.

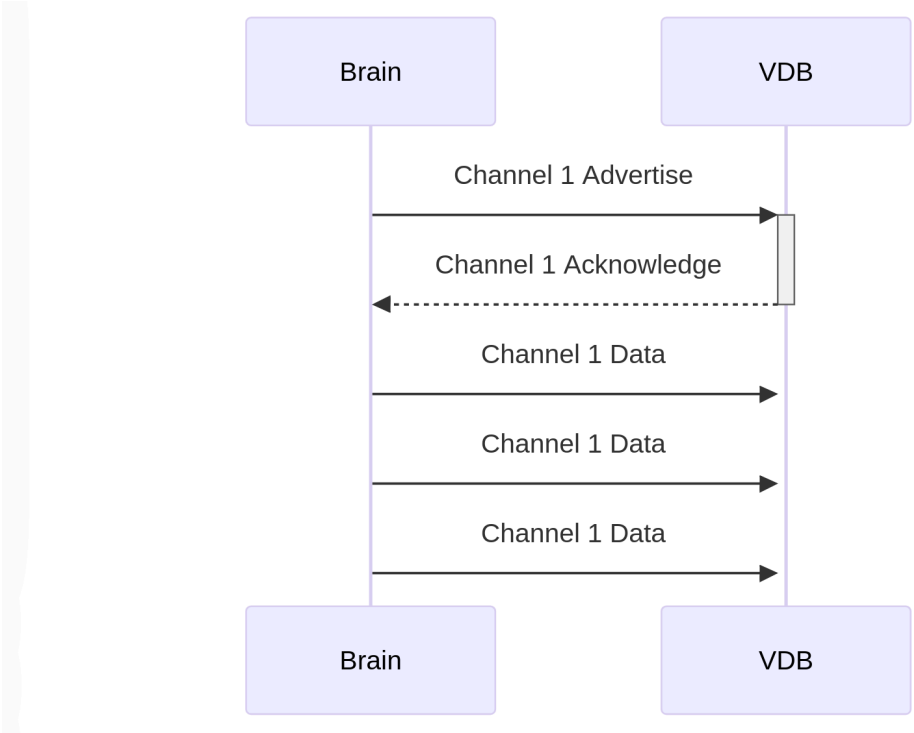
Since the Brain will be talking the most, we have decided that it should be the controller with the VDB only ever "talking" when asked to.

This makes the most sense when considering the types of data either side will send. The Brain needs to send rapidly updating, time series data while the VDB needs to only respond accordingly or send comparatively infrequent updates to control parameters such as PID tuning constants.

For VDB information such as these parameters, we plan to adopt a polling model where the Brain will ask for any available information and know to listen for a response with data or a response indicating no data. At times when it is not expecting a response from the VDB, the Brain is free to send data as it pleases. We are also considering requiring Positive and Negative acknowledgements when receiving data from the VDB as data such as tuning parameters should not be dropped while 100hz time series data coming from the brain can be dropped with little concern. As polling for VDB data is not a priority right now, this has not been decided yet.

Some examples of this strategy are shown below:

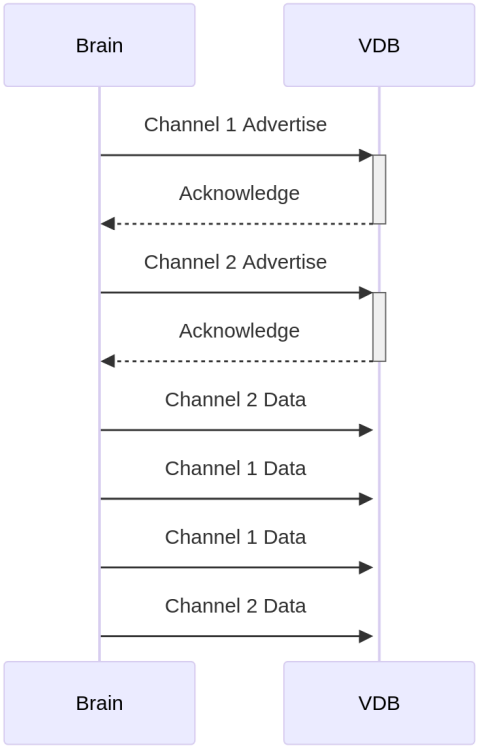
Advertising Channel and Sending Data



Acknowledgement rules

Since only some parts of this protocol requires acknowledgements, we must consider how to know what to acknowledge. If the VDB receives a garbled packet, it can not know what type of packet it is and as such does not know if it needs to acknowledge it. As a rule, the VDB will only acknowledge a packet if its type is Advertise *and* its checksum passes

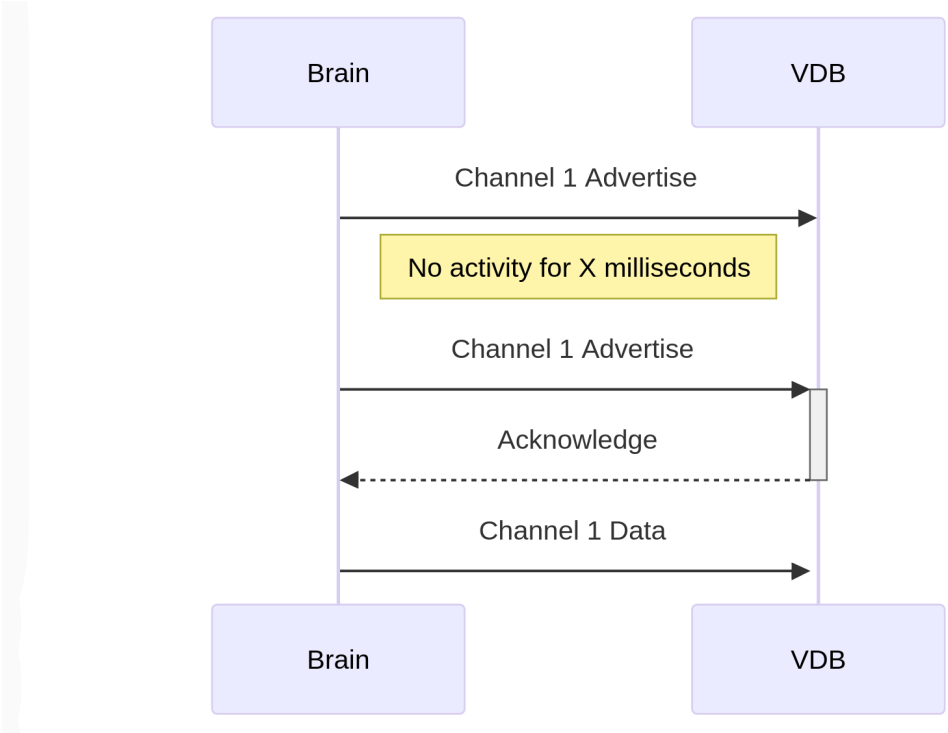
Advertising Multiple Channels



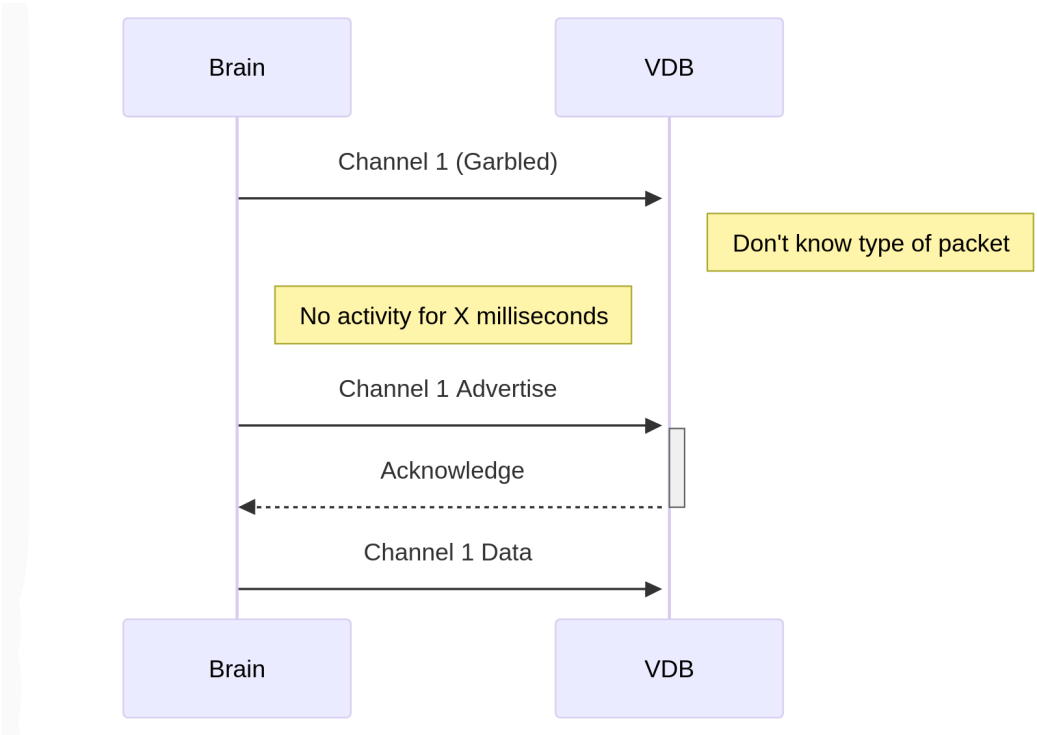
In Flight Limit

In order to simplify the protocol. Only one Advertise/Acknowledge can be "In Flight" at a time. If you wish to advertise 2 channels, you must first advertise channel 1, wait for an acknowledgement, then advertise channel 2

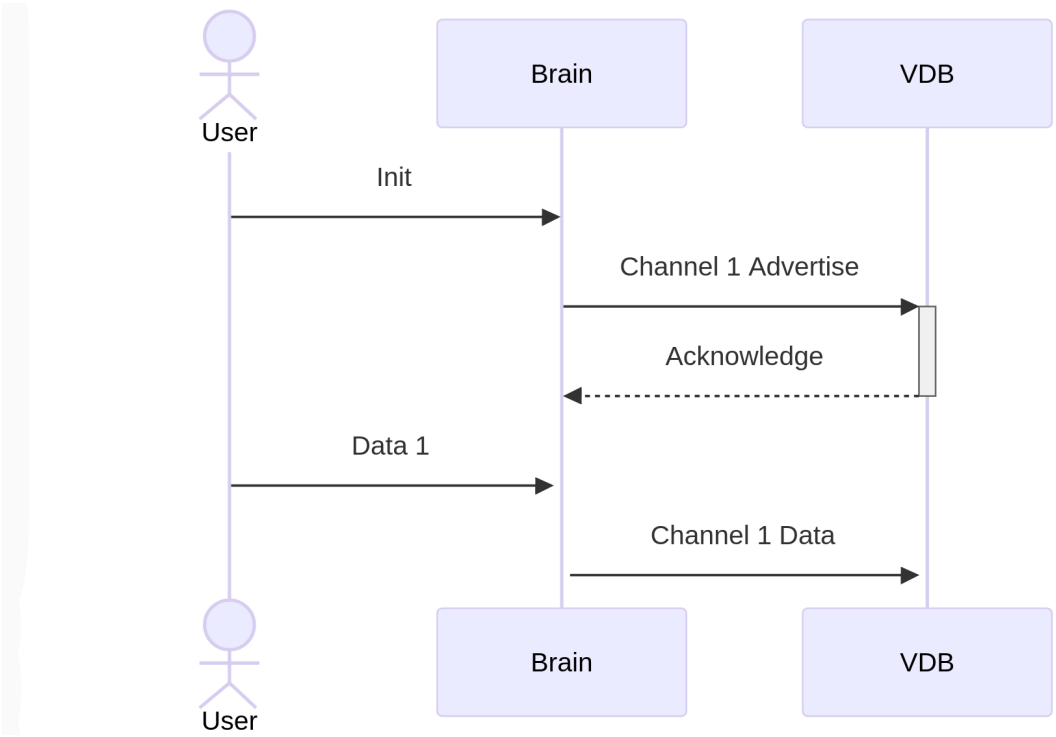
Advertise Timeout



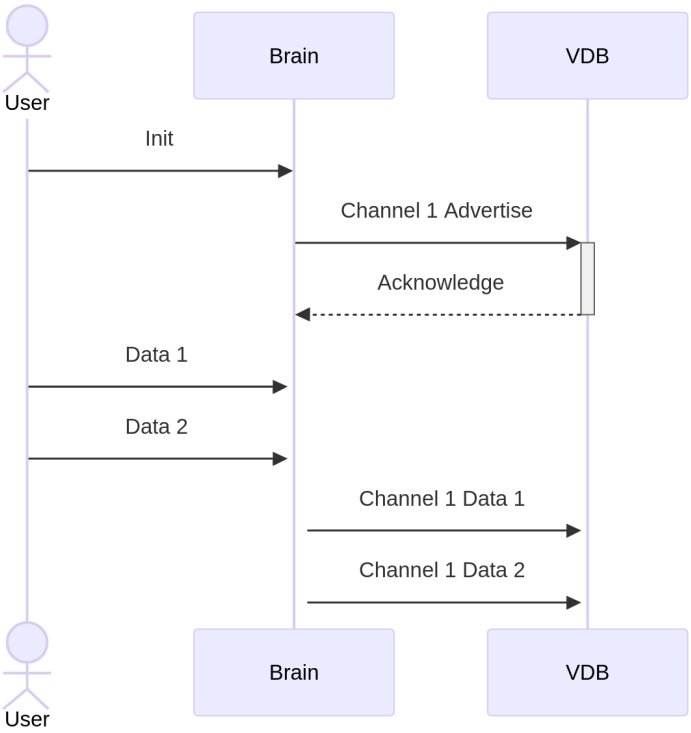
Advertise Failure



In a Context


User
Brain
VDB
Init
Channel 1 Advertise
Acknowledge
Data 1
Channel 1 Data
User
Brain
VDB

Buffering User Data



Brain Connection

While the Vex V5 Platform offers a simple, easy to use electrical system optimized for the uses of the competition, it can be restrictive when trying to use more complicated electrical components. In past years, our team has sought to use COTS Inertial Measurement Units, LIDAR sensors, the Sparkfun Optical Odometry Sensor and other external sensors. These sensors usually use Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), or RS232 UART and are meant to be integrated into a printed circuit board or connected to General Purpose Input Output (GPIO) pins of a microcontroller. However, the V5 platform does not aim to integrate with such devices and as such provides no mechanisms to communicate on these protocols.

The default Vex hardware provides only the following connections: A battery connector, a usb port, and SD Card slot, 21 Smart Ports, and 8 Three wire ports. For the purposes of external communications, only the USB port, smart ports, and three wire ports are of any use. Unfortunately, none of these fit our needs for connecting with other sensors. While the USB port offers a serial connection, it would be playing double duty as the connection to sensors as well as a way to upload code and view output of the program. Additionally, a USB host device would be needed for this connection giving some restrictions about what types of devices we could use here.

The three wire ports on a Vex Brain

The three wire ports are a simple input-output system but are limited in their uses. Using them, we would have to build some protocol to communicate to any device on the other end as the Vex SDK does not offer any standard protocols for communicating complex data over these ports. Additionally, bit-banging any sort of protocol over these outputs will be slow. As well, our team commonly uses almost all or all of these ports for encoders, pneumatics, or potentiometers so we would lose some design flexibility if we chose to use one or more of these outputs for communicating to external devices.

A "Smart cable" connected to a Vex Motor

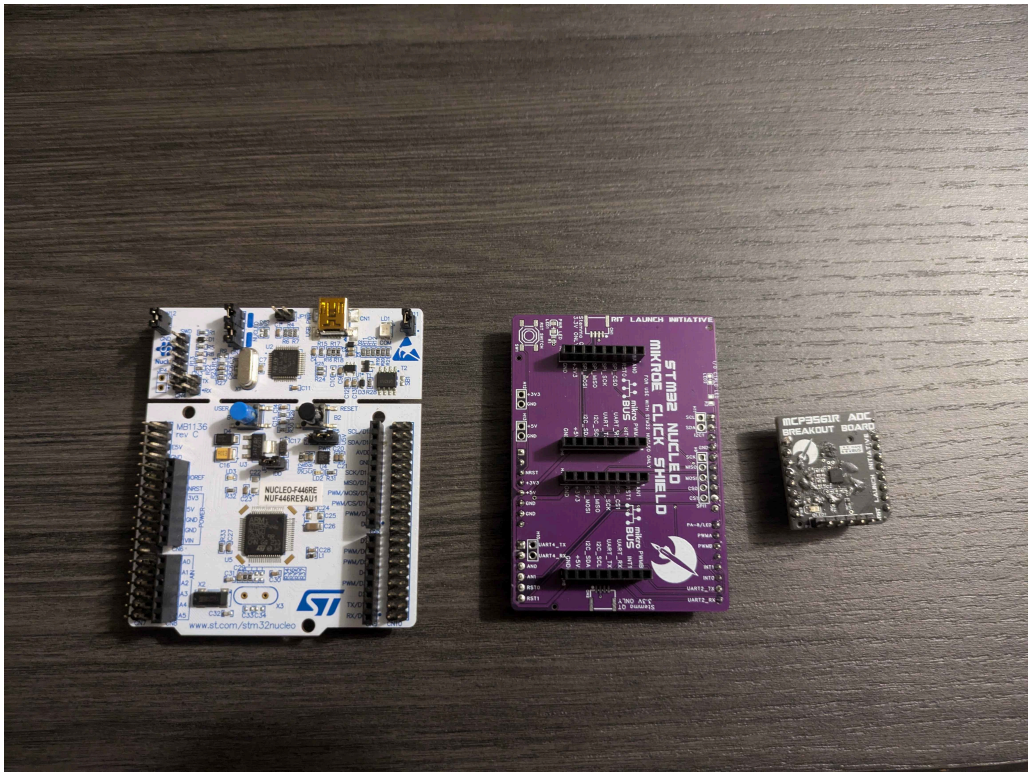
This leaves the smart ports. These are the obvious solution to our problems as we often have extra left open and they provide a standard serial protocol with which to communicate. Unfortunately, not many off the shelf sensors that we would like to use have an RJ9 connector using RS-485 to interact with the brain in this way.

Sensor Connection

One approach would be to create custom PCBs with the sensor, a microcontroller to handle talking to the Brain, and circuitry to handle power over the smart port. While this would be feasible and give us the external devices we so desparately want, it comes at a steep cost. For each sensor we would have to design, layout, and order a new board - at least 50% of which would be shared with other sensor boards.

After some brainstorming, and some inter-team espionage with other RIT clubs, we found the mikroBUS standard. We came across this idea as RIT Launch Initiative, a rocketry team with which we share space and members, uses a single board with mikroBus ports to test development of software using a common platform with new sensor chips. This allows them to quickly design and manufacture small, simple PCBs with just the sensor chip while the larger, complicated circuitry can be shared for a wide variety of sensors over different protocols.

A shield allowing mikro bus connections for a nucleo_f446re

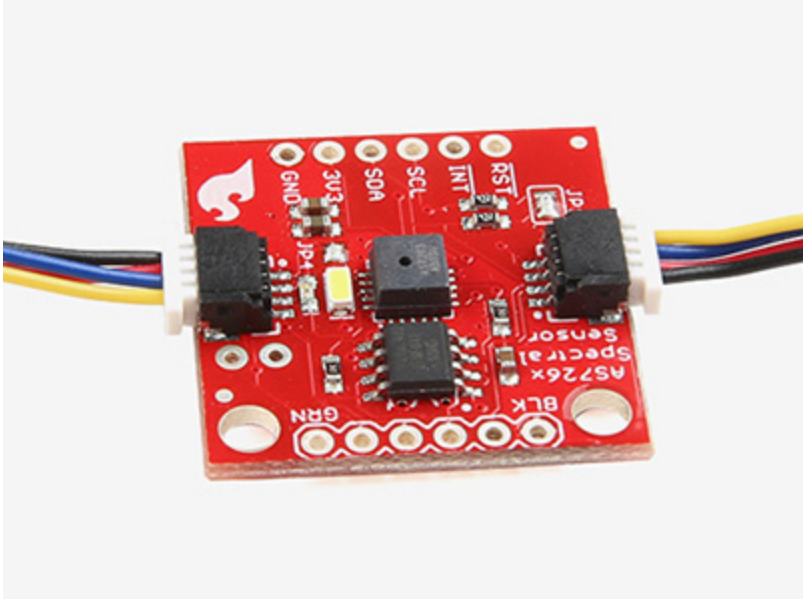


The component parts of the above stack. When a new sensor is needed, only the smallest board must be changed to add the new chip

The pinout for the Mikro Bus connection. Features many common communication standards allowing a wide variety of uses with no change to the underlying connection.

Using this developed standard offers a hardware and software connection to target without having to re-design comparatively complex PCBs after the initial one. Additionally, there is a large library of COTS Mikro Bus compatible boards to choose from - further limiting the number of PCBs we have to design in house. While we are very interested in PCB design, we lack the knowledge, bandwidth, and money to be able to create PCBs each year.

Additionally, for extra flexibility in COTS sensors, we would like to include a Qwiic compatible connector on the board.



A Qwiic connected board

The Qwiic interface is a 4 pin connector that carries power, ground, as well as SCA and SCL to form a extendable I2C interface to external boards. This not only allows us to mount the interface board in one place and a sensor somewhere else (possibly a necessity or orientation dependent sensors such as an IMU) but also allows many sensors over a single output of the board. Where I2C is a bus rather than a point-to-point system, multiple external boards can be chained together and all be accessible.

Continued in **Processor Selection** (Pg. 53)

External Notebook Generator Update 1

Focus: **Notebook Support Software**

By: **Richie Sommers** Date: **10/02/2024**

Continued from **Internal Notebook Generator** (Pg. 22)

%%The first half of this might make more sense to be a different entry where we identify%%

Current Issues

After some time working with the internal, Dataview based Notebook generator, we found some issues.

- Page Numbers: By default, Obsidian does not export PDFs with page numbers. There are some extensions that augment the default behavior, but they did not play nicely with the Dataview rendering
- Table of Contents: A table of contents is vital for understanding the final output. Obsidian does have some extensions that add these, but only to "normal" documents which does not play nice with our Dataview collection.

Luckily, because we chose Obsidian, all the content we have is simply text files. There is no "walled garden" like there are in many other tech products where the only way to do something is how the owning company wants you to.

Not only does Obsidian store its data as normal files on a hard drive, it is stored in Markdown, a well known and well used format with a lot of great tooling.

A next iteration of the Notebook Generator strives to accomplish the following:

- Collect, order, and render notebook entries
- Add page numbers to each page
- Add a table of contents
- Be automatable - an Obsidian PDF export must be kicked off by a user. If we can make this automatic, we can have an up to date preview of the notebook available at any given time

New Solutions

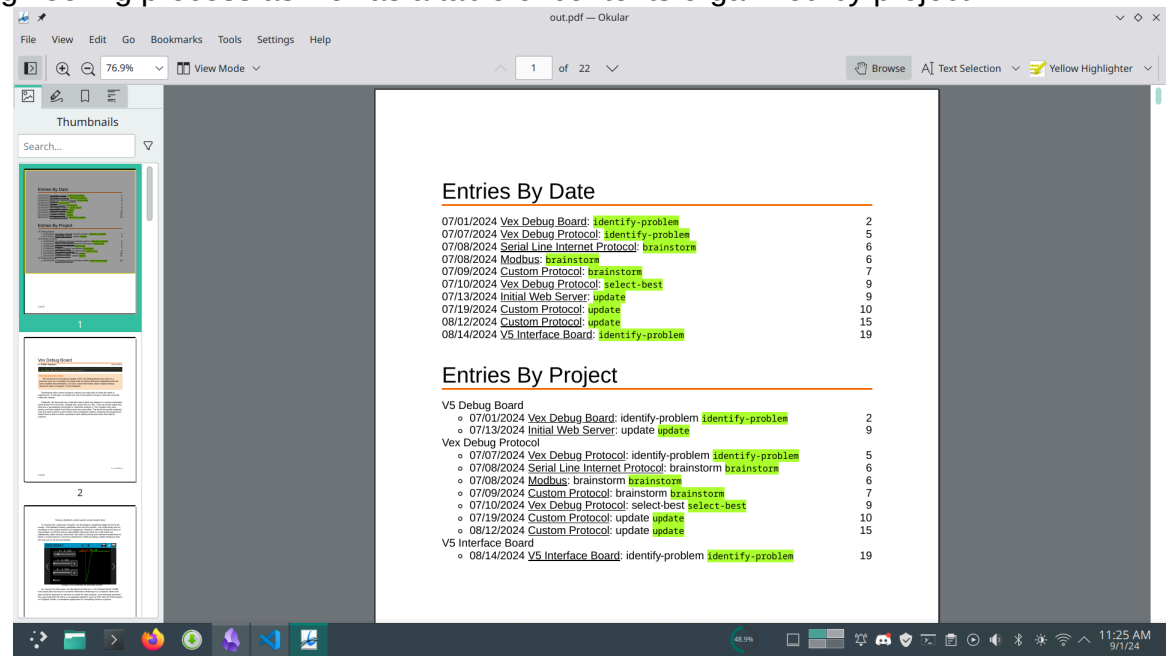
The solution that was developed uses the programming language Go, in combination with the Goldmark markdown parsing library to read and organize entries.

Go was chosen at it is an easy to learn and use language while also being quite fast and low overhead. If we want to be able to create an automated, updating final preview, it can not take a long time to generate the notebook.. However, in comparison to other

high performance competitors such as C or C++, Go is much easier to write. Additionally, the use of external packages is made much easier as Go has a built-in dependency manager.

Goldmark, the markdown parsing library, was chosen for its high quality and ease of extensibility. Goldmark is the library behind Hugo, a web framework that turns markdown files into a static site and is very well maintained. Additionally, Goldmark provides an easy to extend markdown parser allowing for custom elements to be parsed, processed, and rendered. This extensibility is very important as while Obsidian can be used with completely standards-compliant markdown, there are some constructs that (LaTeX formatted equations, YAML frontmatter, and more) , while not part of standard markdown, provide such a benefit to the notebook that this tool must be able to support them.

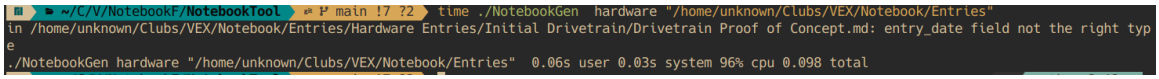
Once we have parsed the markdown files, we organize the entries in order and by frontmatter. Then, we render this output to an HTML document. Included in this document are a table of contents in order of entry date for a chronological view of our engineering process as well as a table of contents organized by project.



Though there are multiple output types that we could render to, we chose HTML as it's formatting capabilities are vast and there are many more members with HTML/CSS experience than there are with experience in advanced LaTeX documentation preparation. As Obsidian is also constructed with HTML and themed using CSS, we can have our editor match the final product closely.

While there is a draft web standard for paged media(CSS Paged Media Module Level 3) by the World Wide Web Consortium, it is not widely supported by browsers. So, a project known as paged.js offers a polyfill (a picece of code used to provide modern functionality to old browsers that do not natively support it) that enables this behavior. In

the future as the standard is finalized and browser support increases, much of the work we put in using paged.js should transfer over.



A screenshot of generating the notebook

The tool has a command line interface where a user specifies the folder where the notebook is as well as which notebook to generate. It will then parse the entries, warn about any malformed entries, and output an HTML file. It runs in less than one tenth of a second.

Continued in **Notebook Backup** (Pg. 60)

Processor Selection

Focus: **V5 Interface Board**

By: **Richie Sommers**

Date: **10/15/2024**

Continued from **V5 Interface Board** (Pg. 45)

The processor is the heart the interface board. In order to achieve its goal of allowing connectivity to external sensors not compatible with the Vex Smart port, it needs some processing power to convert these protocols.

The processor will be selected based on the following criteria:

Compute Performance

Good compute performance allows the interface board to perform intensive calculations alongside its role as an IO expander. For senors such as a LIDAR system, we would like to perform as much of the calculation as possible on the Interface Board. We would like this so that we do not have to funnel *all* of the LIDAR data through a single smart port at 115200 baud. Rather, we would like to do some processing on the Interface Board before passing the pose estimate (a much smaller amount of data than the hundreds to thousands of samples that go into creating that estimate) to the Brain for further use.

Ease of Programming

While learning different programming environments is great fun and very educational, there is a strong pressure from a specific subteam that the software subteam's time is used in a productive way towards the goal of scoring points and winning matches.

With that in mind, as well as lessons learned from the difficulties of onboarding new members to programming V5 Debug Board in the more powerful, but more difficult, ESP Integrated Development Framework, we would like to choose a board with a more simple programming environment. Specifically, many incoming members have experience with the Arduino software development environment. So, a board that offers a similar interface, or an interface with similar levels of beginner-friendliness, is a goal.

Ease of Electrical Engineering

Unfortunately, the knowledge, bandwidth, and money needed to design and build complicated boards is not present on this team (for now). So, a processor solution that allows for a smaller, less complicated board is optimal.

Hardware Compatibility

As a baseline, we require IO connections to a mikroBUS interface, a Qwiic interface, as well as the the ability, through an UART-RS485 bridge chip, to communicate with the Brain over the Smart Ports. If the processor can not handle all this IO, it is not suitable for our needs.

Continued in **Processor - RP2040** (Pg. 56)

Processor - the teensy one

Focus: **V5 Interface Board**

By: **Richie Sommers**

Date: **10/16/2024**

Continued from **Processor - STM32H7** (Pg. 58)

%% How's the NXP teensy 4.1 chip? Has FPU, 600mhz and slightly cheaper

I mean I've been using it for testing and it's worked out very well

like, fast enough to run the ukf at the speed we need, only downside is the fpu is 32 bit so doubles take twice as long %%

Continued in **Changes from V5 Debug Board** (Pg. 59)

Processor - RP2040

Focus: **V5 Interface Board**

By: **Richie Sommers**

Date: **10/16/2024**

Continued from **Processor Selection** (Pg. 53)

The

Continued in **Processor - RP2350** (Pg. 57)

Processor - RP2350

Focus: **V5 Interface Board**

By: **Richie Sommers**

Date: **10/16/2024**

Continued from **Processor - RP2040** (Pg. 56)

%% https://www.sparkfun.com/products/24870 %%

Continued in **Processor - STM32H7** (Pg. 58)

Processor - STM32H7

Focus: **V5 Interface Board**

By: **Richie Sommers**

Date: **10/16/2024**

Continued from **Processor - RP2350** (Pg. 57)

%% https://www.mouser.com/ProductDetail/STMicroelectronics/STM32H743VGT6?qs=%252B6g0mu59x7JjCCaXtUod8g%3D%3D %%

after doing some more searching I'd say this chip is probably the most ideal, has enough flash has enough ram is fast enough has double precision fpu and isn't bga

Relevant Features

- Double Precision Floating Point Unit:

Continued in **Processor - the teensy one** (Pg. 55)

Changes from V5 Debug Board

Focus: **V5 Interface Board**

By: **Richie Sommers**

Date: **10/18/2024**

Continued from **Processor - the teensy one** (Pg. 55)

- Doesn't burn your hand off
- Different Chip
- More GPIO
- More pleasant programming environment

Notebook Backup

Focus: **Notebook Support Software**

By: **Richie Sommers**

Date: **10/19/2024**

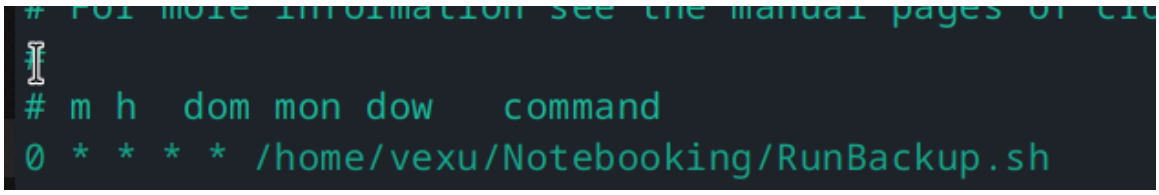
Continued from **External Notebook Generator Update 1** (Pg. 50)

Source control is of vital importance in any software project. While not software, the notebook entries are text (markdown) files that evolve and update much like code files. Also like software, losing work can be devastating and cost hours, days, or even months.

Learning from the lessons and work of the software industry, we adopted Git as a method for backing up the notebook. Since we use it for it's project planning features team-wide and its traditional Git support for the software subteam, all members already have a GitHub account if they need to view the backup.

The actual implementation of the Git backup was fairly straightforward.

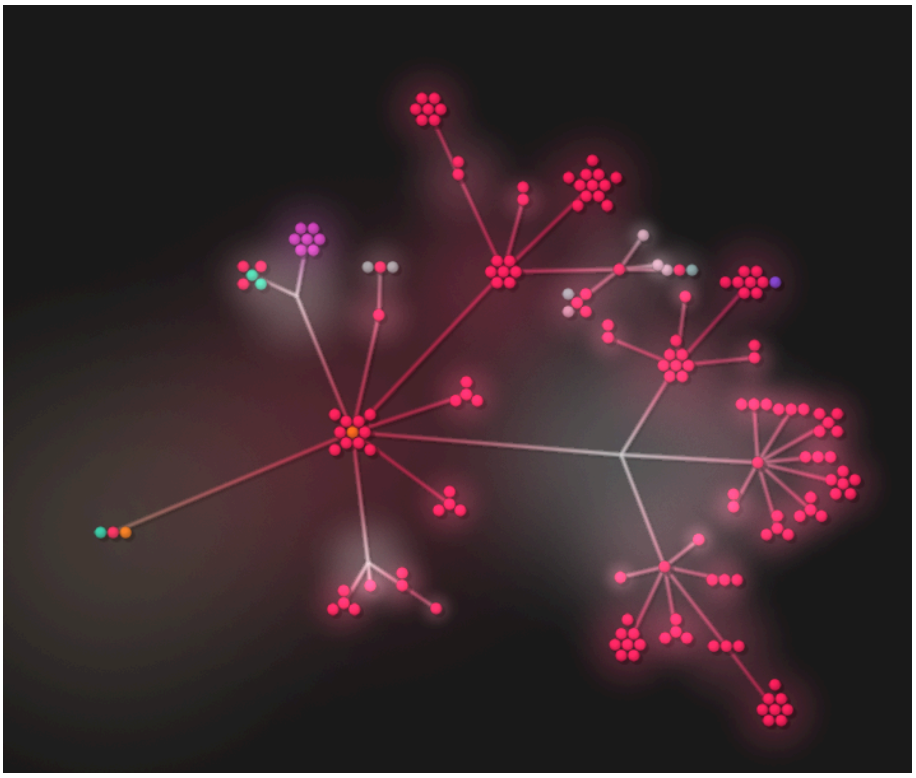
The team was in possession of a Raspberry Pi single board computer used for hosting competitions which acted as the server to run the back up. Google Drive API keys and Git ssh keys were created for the machine. Then, a bash script was setup to run the backup and a cron job used to schedule run it every hour, on the hour.



Crontab for running the backup hourly

The bash script uses `rclone` (a tool for managing remote filesystems on Linux) to copy files from the team's Google Drive to a local folder. Then, the script adds the changes to Git, creates a commit note detailing the time of the backup, logs its activity, then pushes the changes to the Git server hosted by GitHub.

While we do not expect to require the backup, it is there in case we do.



A visualization of the backup Git Repository as of 10/22/24. Created using gource

Continued in **Notebook Backup Fix** (Pg. 78)

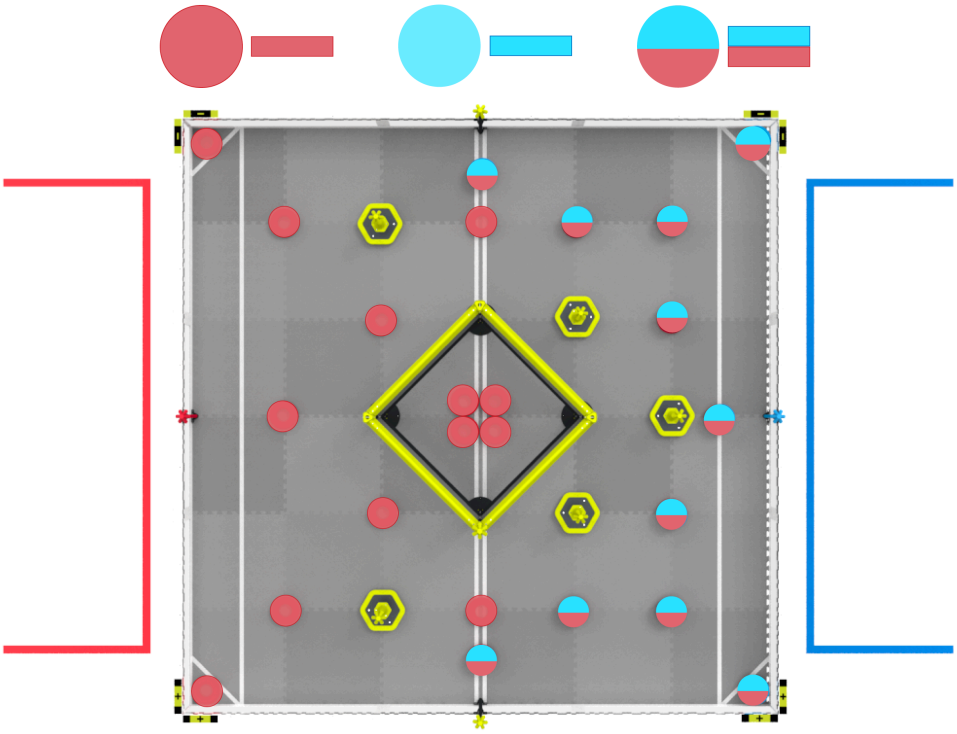
Autonomous Skills Path

Focus: **Robot Skills**

By: **Victor Rabinovich**

Date: **10/20/2024**

Part of what software needs to do is create autonomous path for the Skills challenge portion of the VEX U competition. As part of our onboarding process, and part of preparing for skills comp in late November or Mid December, the software sub team split into three groups and came up with a few different autonomous skills paths. The paths are aimed at scoring the maximum amount of points given the resources and robots we have to use.



The VEX U skills field setup

Continued in **Skills Path Group 1** (Pg. 68)

Skills Path Group 3

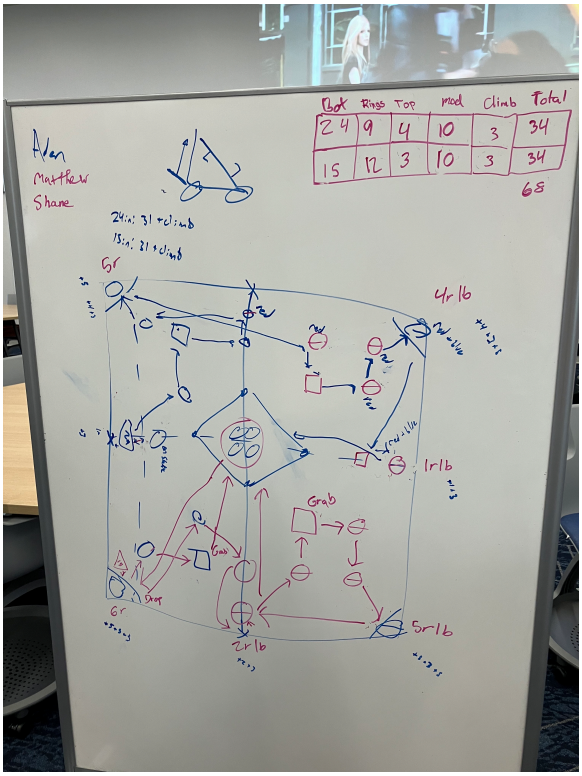
Focus: **Robot Skills**

By: **Victor Rabinovich**

Date: **10/20/2024**

Continued from **Skills Path Group 2** (Pg. 65)

We split the board into two parts, the top half and the bottom half with the 24 inch bot in the top section and the 15 inch bot in the bottom section. Our goal is to get all the red rings on first, then get a few blue rings afterwards and then finally do a low climb on both. Our bot assumes it can hold two rings, hold mobile stakes, and can score on ally and neutral stakes.



The 24 inch bot starts on the horizontal middle line and scores the ring in front of it first on the ally stake. It then heads up to grab another ring and the mobile stake to then score on that stake, the bot continues to hold that stake and moves towards the vertical middle line to score the two red rings on the line on the mobile stake. The bot then goes towards the corner to score the two red rings on the mobile stake while leaving the mobile stake in the corner. The bot then goes past the vertical middle line to then grab the nearest red ring and score it on the mobile stake. The bot grabs the mobile stake and scores the 3 red rings near it and the blue ring in the corner on the held mobile stake and deposits the mobile stake in the corner. The bot then scores the red and blue ring on the horizontal middle line on the nearby mobile stake and heads towards the ladder to do a low climb.

The 15 inch bot starts right next to the bottom corner to score the two nearby rings on the mobile stake nearby. The 15 inch bot grabs the mobile stake and brings it to the ladder to score the 4 red rings in the ladder to then bring it to the corner. The bot will

then grab the remaining red ring before heading to the vertical middle line to grab the solo red ring and score them on the neutral stake. The bot will then grab the remaining red ring and the closet red ring to then score it on the mobile stake. The bot will then grab the mobile stake to them score the two red rings nearby. The bot then grabs the two rings in the corner and score then, the blue one last and then leaves the mobile stake in the corner. The bot then goes back to the vertical middle line, scores the blue ring on the neutral stake and then heads to the ladder to do a low climb.

This path has a lot of distance to cover in a short amount of time, it might be hard to get all the red rings to then score the blue rings. We are making assumptions about how fast the robot can move and how fast the robot can score rings.

Scoring:

We expect this path to get us 68 points, most of which are from top rings and corner stakes.

Skills Path Group 2

Focus: **Robot Skills**

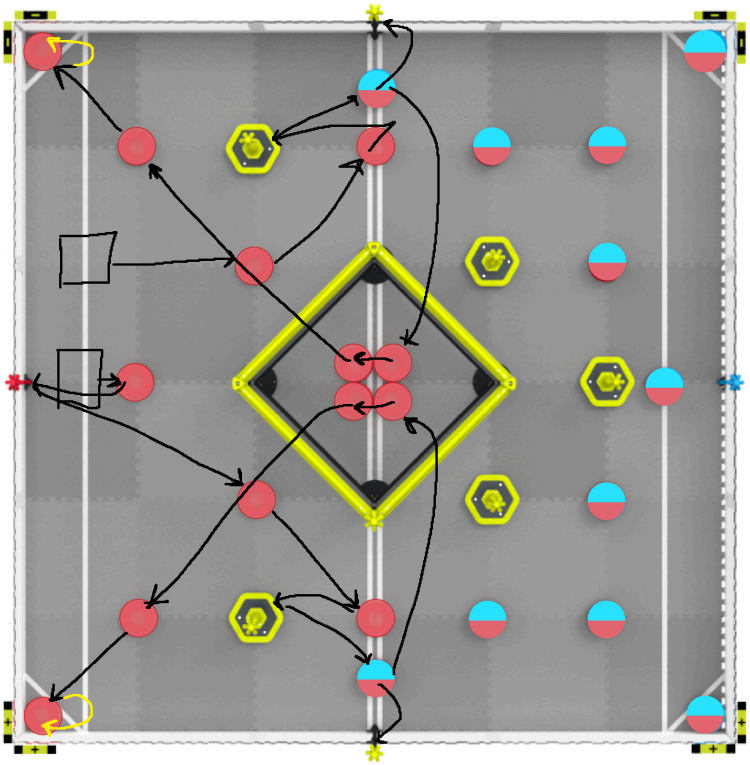
By: **Victor Rabinovich**

Date: **10/20/2024**

Continued from **Skills Path Group 1** (Pg. 68)

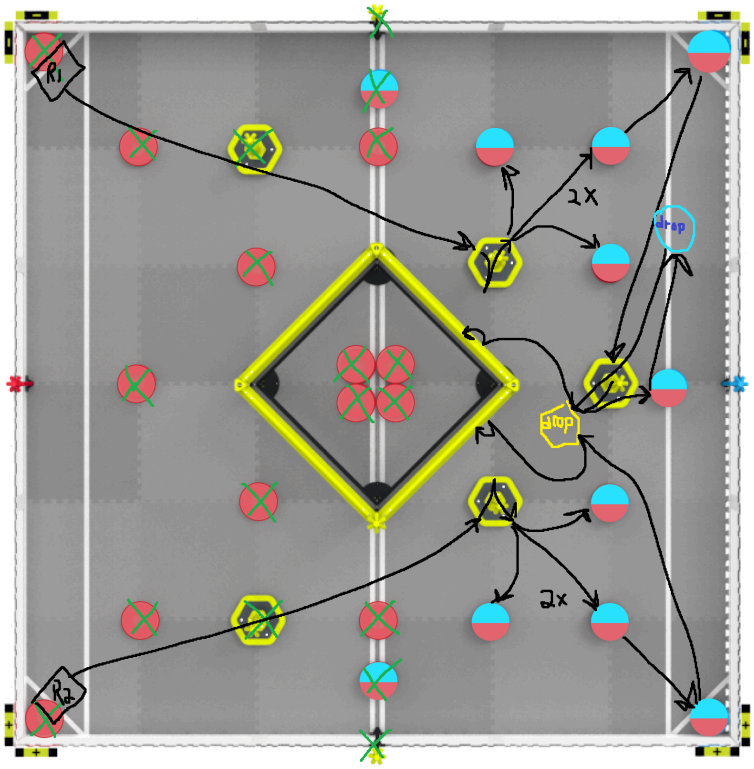
This skills path effectively takes place over two phases, the first phase involves the left half, which is far less complex, and the right half, which is entirely stacked rings and is more complex. Requirements for this path to be possible (in this configuration) are the ability to score on alliance stakes while a mobile goal is held, and would most likely require a vision system that can determine positions of rings and mobile goals relative to the robot.

This path attempts to maximize points scored by filling all mobile goals, placing four of them into corners, filling the two neutral stakes, and placing one ring on the red alliance stake. After the rings are finished being scored, both robots will attempt to climb as high as possible in the remaining time.



First Phase: One robot starts the match inline with a ring and immediately places it on the alliance stake. This robot will remain behind for the rest of the match, which is critical for the second phase. After that, both robots complete the same path, collecting two red rings and a mobile goal, scoring both of them immediately. They then pick up the red then the blue rings next to the neutral stakes, and score them on the neutral stakes in that order. Then they move under the ladder to pick up and score two more red rings on the mobile goal. Since one robot is moving more slowly, they will reach the middle at different times, and will not interfere with each other. If the rings that the first

robot does not pick up are pushed, a vision system, or slightly altered path might be required for the second robot to pick up its rings. After this the robots move toward the corners, and pick up the last two rings, score them turn around and drop their mobile goals, completing the first phase.



Second Phase: Both robots start the second phase in the red side corners. They immediately move to the other side of the field, and pick up the closest mobile goals. A series of six back and forth movements are required to score the 3 stacks of red and blue rings, since blue rings only count if they're scored above red rings on a stake. Both robots move to the corner, pick up the red then blue rings, and drop their mobile goals. At this point, the first robot is still ahead, so it will pick up the final mobile goal first. On its path to the mobile goal it drops its blue ring in a set location. When it picks up the mobile goal it will score its red ring, then pick up the stack next to it, but only score the red ring. Robot 1 will then drop the mobile goal to go retrieve the blue ring that it dropped before. While Robot 1 is doing this, Robot 2 will grab the mobile goal, and immediately score both its red and blue rings, then immediately drop the mobile goal and move to climb the ladder. Once Robot 2 has dropped the mobile goal, Robot 1 grabs it again, and scores both of its blue rings. Finally, Robot 1 drops the mobile goal, since there are no remaining corners, and moves to the ladder to climb.

This path is ambitious, but it would maximize ring scoring, and provide the possibility of a climb, while also being fast. Various assumptions about the robot's mechanisms are required, and significant effort will be required for programming, including vision and very accurate odometry.

Scoring:

Point Source	Points
Alliance Stake	3
Neutral Stakes	8
Mobile Goals	40
Mobile Goals in Corners	20
Climb	0-24
TOTAL	71-95

Continued in **Skills Path Group 3** (Pg. 63)

Skills Path Group 1

Focus: **Robot Skills**

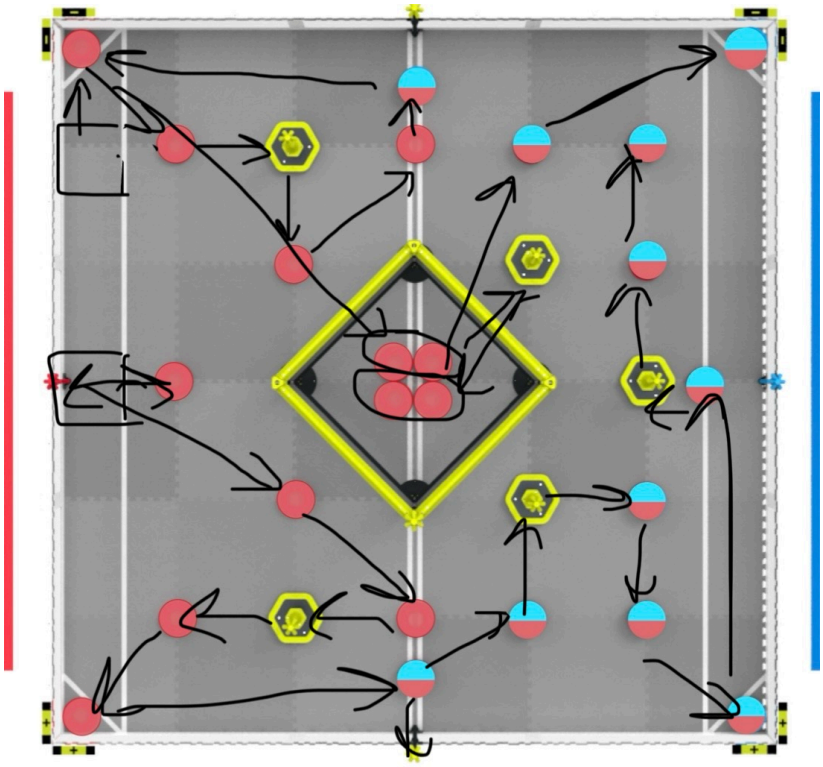
By: **Victor Rabinovich**

Date: **10/20/2024**

Continued from **Autonomous Skills Path** (Pg. 62)

This skill path splits the field into top and bottom and assumes that both robot has the ability to: hold on to a mobile stake, hold two rings at a time, able to pick just the bottom ring (if its a stack), score the ring onto the carried mobile stake, score the ring onto a neutral stake while carrying a mobile stake. The main goal of the paths was to prioritizing scoring all red rings (and ignore a few blue rings for less risk) through straight paths. We want to make sure that all the red rings are below the blue due to RSC5.

<RSC5> Any red Ring Scored above a blue Ring on the same Stake will not have a point value.



Our path for the top robot is to first go north, pickup the ring, go southeast, pick up the ring, go east, and then score it onto the mobile stake. While carrying the stake, it goes south, northeast, north while picking up all the rings and drop off the stake (from top to bottom: BRRRRR) at the top left corner. It then go southeast to center and pick up two rings, go northeast, place on mobile stake, pick up mobile stake, go southwest, score two rings, and then go up and only pick up the bottom red ring. Then it goes to the top right corner, picking up on the bottom red ring, score it on the mobile stake (RRRRRR) and leave it at that corner. Lastly it goes straight to the ladder to begin its climb

For the bottom robot, it starts off in the center picks up the ring and score it on the alliance stake. Then it goes southeast picking up two rings, goes west, score on mobile stake, pick up stake, go west, pick and score ring, go southwest, pick and score ring and leave the stake (RRRR) at the bottom left corner. Then it goes east, picking up two ring and score on the neutral stake. Then it goes northeast, pick up only the bottom red, go north, score ring on mobile stake and pick up stake, go east, pick and score only bottom red, go south, pick and score only bottom red, go south east, pick and score two rings and leave the stake (BRRRR) at the bottom right corner. Then it goes north (here we weren't sure if the bot is allowed to score rings on the blue alliance stake) pick up just the bottom red, score on mobile stake, pick up mobile stake, go north, pick up and score just the bottom red, go north pick up and score two rings(BRRR). Lastly it goes southwest to climb the ladder.

Scoring: Ideal case (all red was picked up and scored):

Point Source	Points
Alliance Stake	3
Neutral Stakes	4
Mobile Goals	35
Mobile Goals in Corners	20
Climb	0-24
TOTAL	62-86

If not all red was scored (Due to VURS4):

Point Source	Points
Alliance Stake	3
Neutral Stakes	1
Mobile Goals	26
Mobile Goals in Corners	20
Climb	0-24
TOTAL	under 50-74

<VURS4> Each blue Ring only has a point value if:

a. All red Rings in the Match have been Scored on Stakes and have point values
b. At least one red Ring is Scored below the blue Ring(s) on that Stake c. No red Rings are Scored above the blue Ring(s) on that Stake

Continued in **Skills Path Group 2** (Pg. 65)

Unscented Kalman Filter

Focus: **Localization**

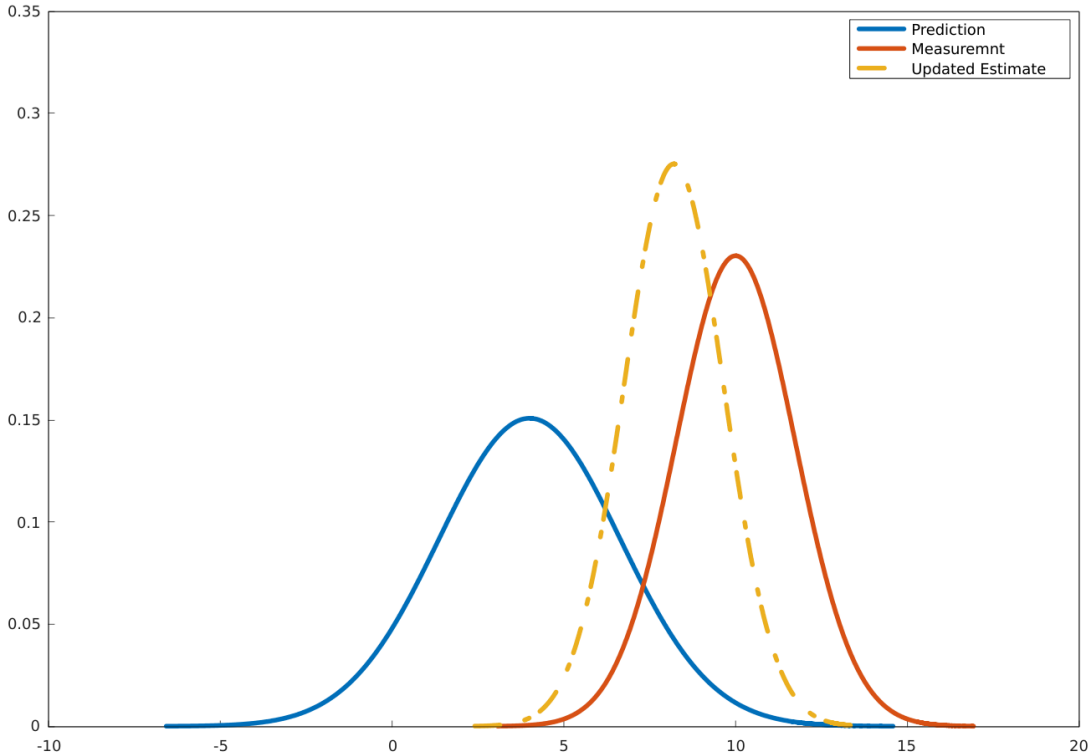
By: **Jack Cammarata**

Date: **10/21/2024**

Continued from **Sensor Fusion** (Pg. 75)

includes background, what a kalman filter is, why sensor fusion is important, our implementation (GENERIC - lots of graphs and such but don't talk about odometry or lidar)

Kalman Filter Background



No sensor exists in real life that is perfectly accurate and precise. While sensors are sometimes "good enough" at providing accurate estimates of real world values, their raw output usually has some amount of error. For example, a thermometer measures temperature, but its measurement might be incorrect by anywhere from 0 to 2 degrees.

A simple way to deal with this would be to take multiple measurements and find their average, but this isn't always necessarily possible or effective, especially if the sensor is very noisy. A moving average is also not ideal, since it introduces considerable lag to the estimate The Kalman filter is the optimal method to estimate some state by combining noisy sensor measurements, and an imperfect model of how the state changes over time.

Many systems operate like this, for example a car may know how fast its wheels are spinning, and thus can use this to predict how far it will move in some amount of time,

simply multiply rate and time. The car may also sometimes receive GPS measurements accurate to within 3m, a Kalman filter will optimally combine the car's wheel speeds and GPS measurements to obtain an estimate that is more certain than if only one sensor was used.

Gaussian Noise

The values that are estimated by the Kalman filter are called the state. In the Kalman filter these mean values are called $\mathbf{x}$, in statistics the mean is μ This is stored as a column vector, however for space reasons this document will show them as a transposed row vector:

$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$$

In this case, the Kalman filter is estimating three variables. The Kalman filter stores each variable as a gaussian, a mean μ and variance σ_x^2. also stores the variance of each variable in the state vector, specifically it stores them in the form of a covariance matrix, so it is also storing the correlation between each variable:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \, \sigma_{xy} & \sigma_y^2 \end{bmatrix}$$

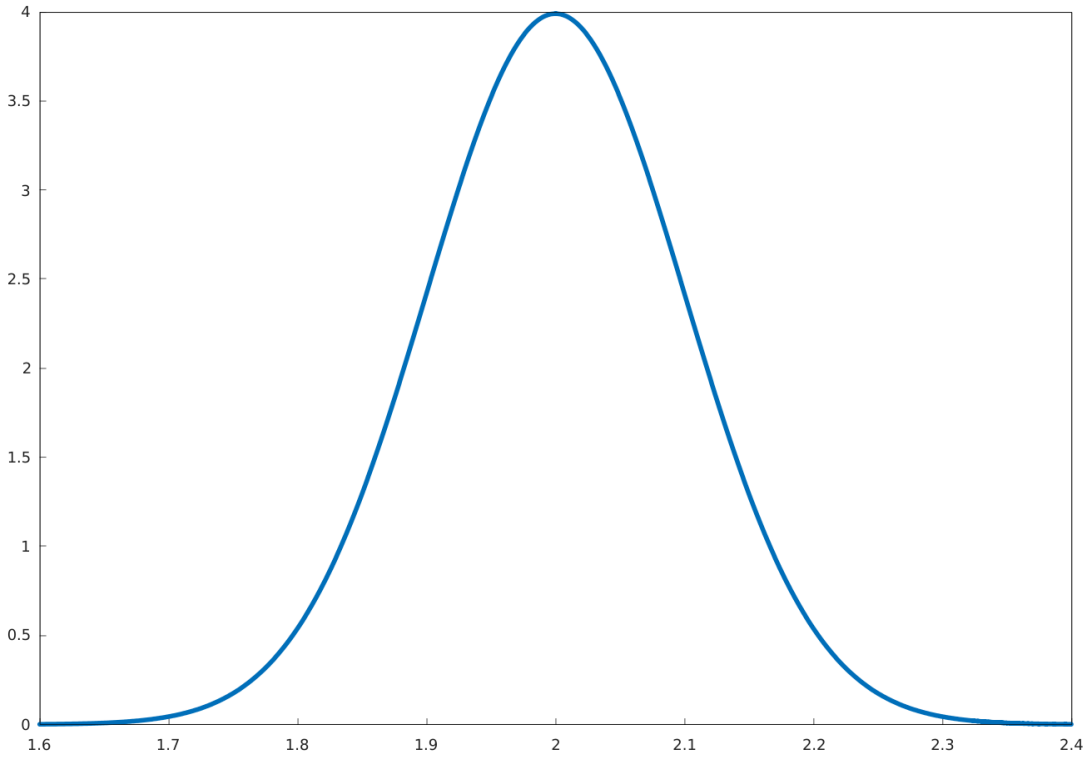
A more common measure of the error is standard deviation, σ, where variance is simply σ^2. The covariance σ_{xy} represents the correlation between x and y, if σ_{xy} is positive, then x and y are linearly positively correlated, so as x increases y increases linearly. if σ_{xy} is negative then x and y are linearly negatively correlated, so as x increases y decreases.

For simplicity, this is a univariate example of a state which only tracks x where $x = 2, \Sigma = 0.01$:

$$\mathbf{x} = \begin{bmatrix} x \end{bmatrix}^{\mathrm{T}}, \mu = \begin{bmatrix} x \end{bmatrix}, \Sigma = \begin{bmatrix} \sigma_x^2 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} 2 \end{bmatrix}^{\mathrm{T}}, \mu = \begin{bmatrix} 2 \end{bmatrix}, \Sigma = \begin{bmatrix} 0.01 \end{bmatrix}$$

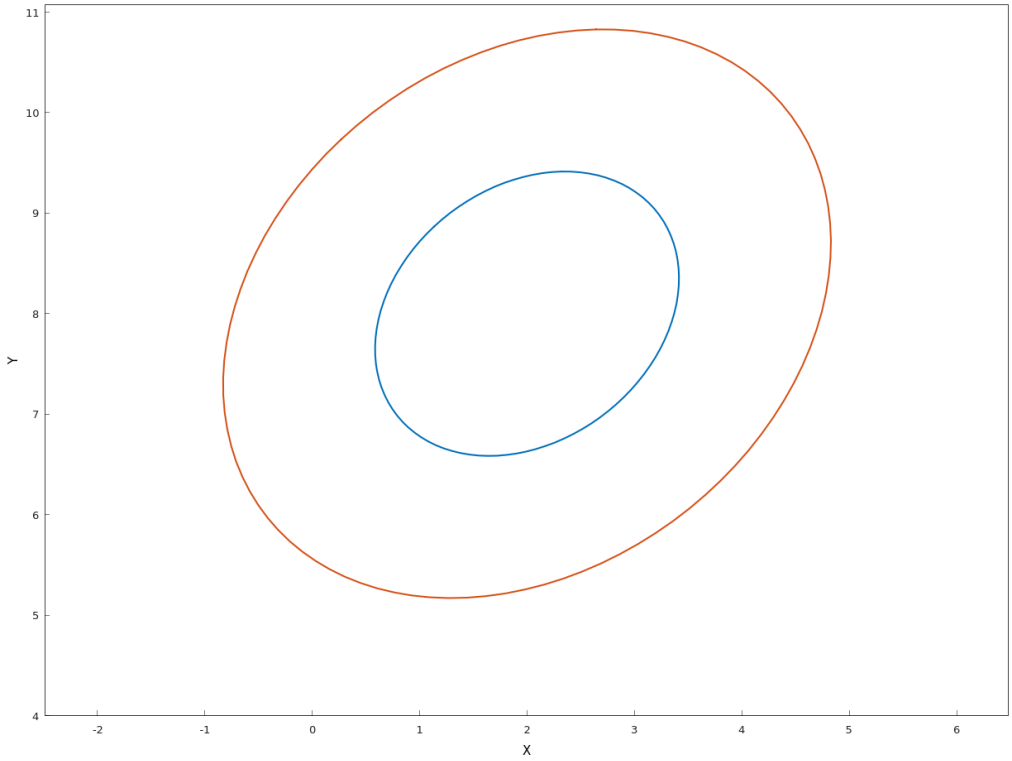
This is the state of only x, and its variance σ_x^2. We can show what this means visually using a graph of the probability density function for this gaussian.



This function is a normal distribution, so it includes the properties of one. The area under a region of the function is the percent chance that a point chosen randomly will be within that range. About 68% of samples will fall within 1 standard deviation of the mean, 95% within 2, and 99.7 within 3.

Extending this plot to two dimensions is simple, plot ellipses at n standard deviations:

$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}, \Sigma = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$



Continued in **Dead Wheel Odometry - IMU** (Pg. 76)

Robot Localization

Focus: **Localization**

By: **Jack Cammarata**

Date: **10/21/2024**

%% discuss specifics of using ukf for robot localization, basically the state vector, F(x) (process model), H(x) (measurement model(s)), all sensors used, pictures from the simulation, maybe include lidar maybe not? I might just give it its own heading in here and its own file or project like for the debug and interface boards /\ | move this elsewhere, to something about developing the chosen solution?

(diagram of field with robot drawn as a square moving across it, tracking pose) %% Robot localization is the process that allows a robot to know its position (pose) on the field. Knowing its pose is critical for a robot to complete an autonomous routine correctly and consistently. Without some localization system, the robot would be unable to navigate around the field dynamically, Localization opens up lots of opportunities for features such as path following, path generation, and overall autonomous accuracy and speed optimization. There are many ways to accomplish this, with varying precision, accuracy, and robustness.

There are two types of tracking, absolute tracking, and relative tracking.

- Absolute tracking tells the robot its pose directly, in a car this would be the GPS, which outputs coordinates on the earth.
- Relative tracking tells the robot its acceleration or velocity, but not its pose, in this case the robot must translate its pose by the measured velocity, and must have a known accurate starting pose. In a car this would be your odometer, measuring distance driven based on wheel rotation, it does not tell you the coordinates of your car, it tells where it is relative to a starting point.

Ideally, a localization system would include both absolute tracking and relative tracking because the absolute tracking would prevent drift in the pose and give a starting pose, while the relative tracking would provide accurate short term poses and speeds. The implementation of such a system is often far more complicated than simply using one or the other.

In the past, RIT has relied only on relative tracking during competitions, so pose drift has remained a relevant issue, especially for long autonomous skills runs. More advanced localization systems will be considered in order to reduce, or potentially entirely solve the pose drift problem.

Continued in **Sensor Fusion** (Pg. 75)

Sensor Fusion

Focus: **Localization**

By: **Jack Cammarata**

Date: **10/21/2024**

Continued from **Robot Localization** (Pg. 74)

Continued in **Unscented Kalman Filter** (Pg. 70)

Dead Wheel Odometry - IMU

Focus: **Localization**

By: **Jack Cammarata**

Date: **10/22/2024**

Continued from **Unscented Kalman Filter** (Pg. 70)

(some picture of something) Dead wheel odometry involves measuring rotation of three wheels placed strategically on the robot in order to determine relative motion of the robot. This system can work on its own, however by using a sensor fusion algorithm to also include data from an IMU, the pose estimate, especially the heading, would have improved accuracy.

PROS

- More accurate over time than drive wheel odometry, wheels are less likely to slip and combines multiple sensors
-

CONS

- Relative tracking only, has no way to measure absolute position on the field, so will still be prone to some drift
- Like tracking wheels, it also does not track if the robot is lifted off the ground or travels over bumps, the sensor must be exactly 10mm above the ground to function properly.
- Not extensible, it is not possible to read raw measurement data from the sensor, so this sensor cannot be used effectively with a custom sensor fusion algorithm. This means that there is not a good way to use this sensor along with some absolute sensor to correct drift.
- The team does not have one yet, and it is fairly expensive

Continued in **Drive Wheel Dead Reckoning** (Pg. 79)

SparkFun Optical Tracking Odometry Sensor

Focus: **Localization**

By: **Jack Cammarata** Date: **10/22/2024**

Continued from **Drive Wheel Dead Reckoning** (Pg. 79)

(some picture of something) Recently SparkFun released a product advertising an all in one localization solution consisting of a single board with an onboard IMU and optical sensor. The optical sensor is similar to a mouse sensor, and tracks velocity, the gyroscope in the IMU is used to track rotation, since the sensor cannot on its own.

> **WARNING**
>
> Requires custom interface board to allow brain-sensor communication!

PROS

- All in one solution, it does calculations with an onboard processor and simply outputs a position on the field - custom sensor fusion algorithm is not needed
- Minimal work required for tuning, it runs a routine on startup and simply does it by itself.
- Very small physical footprint, opens up space for hardware compared to other localization solutions.
- Less prone to drift than tracking wheels since it does not have a wheel that can slip

CONS

- Relative tracking only, has no way to measure absolute position on the field, so will still be prone to some drift
- Like tracking wheels, it also does not track if the robot is lifted off the ground or travels over bumps, the sensor must be exactly 10mm above the ground to function properly.
- Not extensible, it is not possible to read raw measurement data from the sensor, so this sensor cannot be used effectively with a custom sensor fusion algorithm. This means that there is not a good way to use this sensor along with some absolute sensor to correct drift.

Continued in **Dead Wheels Update** (Pg. 92)

Notebook Backup Fix

Focus: **Notebook Support Software**

By: **Richie Sommers**

Date: **10/22/2024**

Continued from **Notebook Backup** (Pg. 60)

After a number of days of the Raspberry Pi running as a backup provider, it was realized that it was not keeping the Git repository in sync with the actual state of the notebook.

The original implementation of the script used `rclone copy` to take files from the Google Drive folder and copy to the Git staging directory. However, while this tracked changes and new files, it did not delete old files. This can possibly save a user from accidentally overwriting data. But, we want our local copy to exactly mirror the remote directory - overwriting data is a non-issue as we save our work to GitHub. So we needed to use the `rclone sync` command which synchronizes the remote Google Drive folder and our local copy, deleting as necessary.

Continued in **External Notebook Generator Update 2** (Pg. 84)

Drive Wheel Dead Reckoning

Focus: **Localization**

By: **Jack Cammarata**

Date: **10/22/2024**

Continued from **Dead Wheel Odometry - IMU** (Pg. 76)

(some picture of something) Prior to the start of this season, RIT has relied on a relatively simple system that includes only two types of sensors, encoders and an inertial measurement unit (IMU). The encoders are on the drivetrain's traction wheels measuring their rotation, and the only the gyroscope of the IMU is used to determine the robot's heading. This simply uses the kinematics of a tank drive to calculate how the position will change at each time step.

Since this system has been used for some time now, the team has well developed documentation on the setup, tuning, and use of this system. The code is already written and is part of the Core library, so to use it on a robot is very easy.

PROS

- Since this system is well documented, and the algorithm is already written and has been used effectively for years, it is very easy to set up and use.
- There are minimal constraints for hardware, the encoders are fairly small and are adjacent to the wheels, so they're out of the way of the rest of the mechanisms on the robot.
- Since the team has done this before, there is no need to purchase new hardware.

CONS

- Dead reckoning is not ideal, since it does not include any method of absolute tracking. This means that over time the pose estimate drifts from reality due to many factors such as wheel slip.
- If the robot is perturbed, and/or the wheels slide for any reason, the robot will not detect this at all, and the estimate will not move despite the robot physically moving.

Continued in **SparkFun Optical Tracking Odometry Sensor** (Pg. 77)

Correct Match Scoring Calculator

Focus: **Scoring Calculator**

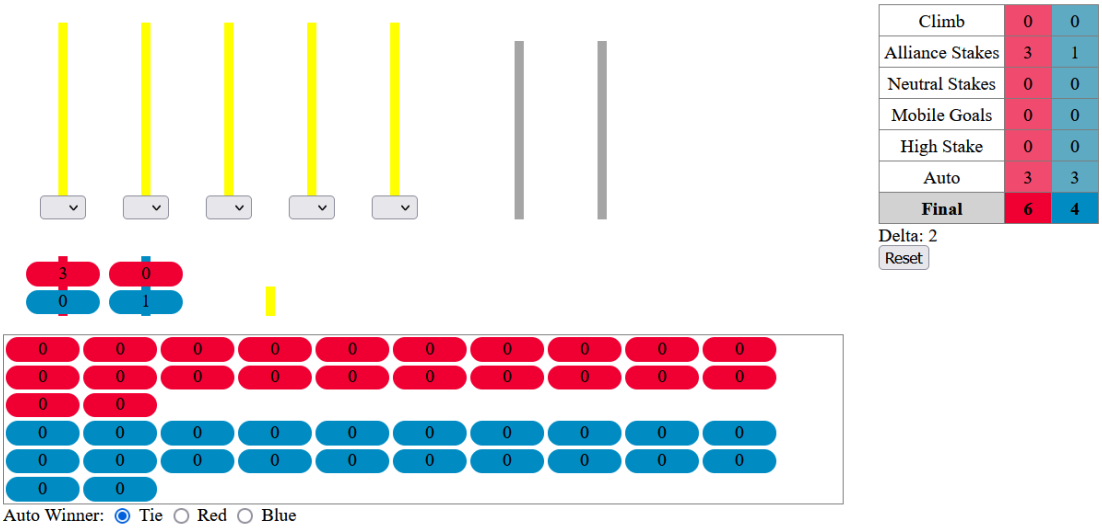
By: **Zoe Rizzo**

Date: **10/26/2024**

Continued from **GitHub Pages Scoring Website** (Pg. 11)

To make sure the scoring calculator website is as intuitive as possible, users should not be able to perform illegal actions. Instead of a message appearing telling the user an action is illegal, they should just not be able to do anything illegal.

The main fixes needed are making sure wrong colored rings on alliance wall stakes are worth 0 points and that only the maximum number of rings can be added to each stake.



Wrong Color Rings Worth 0 Points on Alliance Wall Stakes

In addition to fixing illegal actions, climb points needed to be added to the calculator. The High Stake Bonus also needs to be calculated correctly.

<SC9> A High Stake bonus is available to an Alliance that ends the Match with a Ring Scored on the High Stake. Each Robot from that Alliance which has earned points for a Climb will receive an additional two (2) points for that Climb.

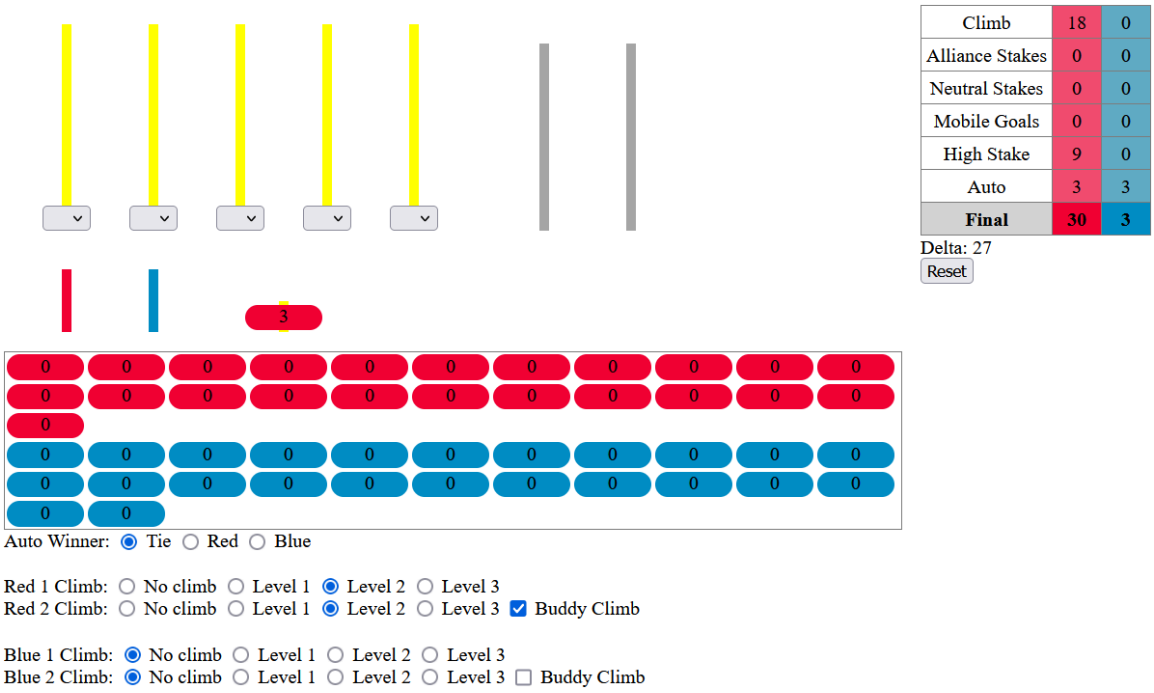
<Q&A 2047> Both the Climb points and the High Stake bonus are doubled under <VUG2b>.

According to <SC9> and Q&A #2049, each robot that earns climb points will receive an additional 2 points if the High Stake is scored. Additionally, if one robot is considered

"buddy climbed," that robot's High Stake bonus will double, earning an additional 4 points.

Definitions

"Buddy climb" is a term for game manual rule <VUG2b>, which explains that a robot that meets all requirements for Climbing, but is not contacting the ladder at the end of the match will earn double Climbing points. The way this is achieved is by having one robot support the other so that one is not contacting the ladder, hence the term "buddy climb."

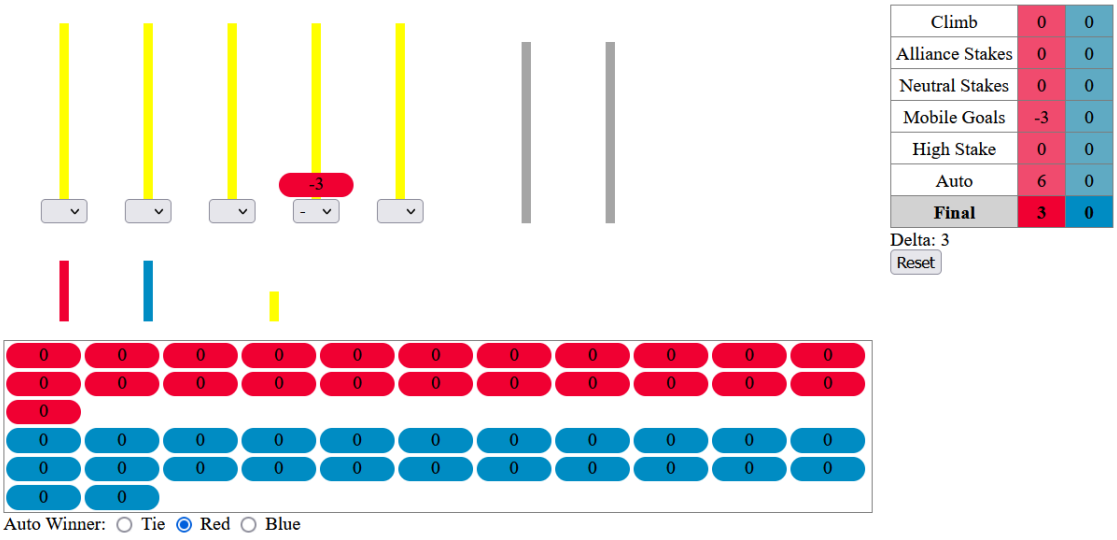


High Stake Bonus with Buddy Climb

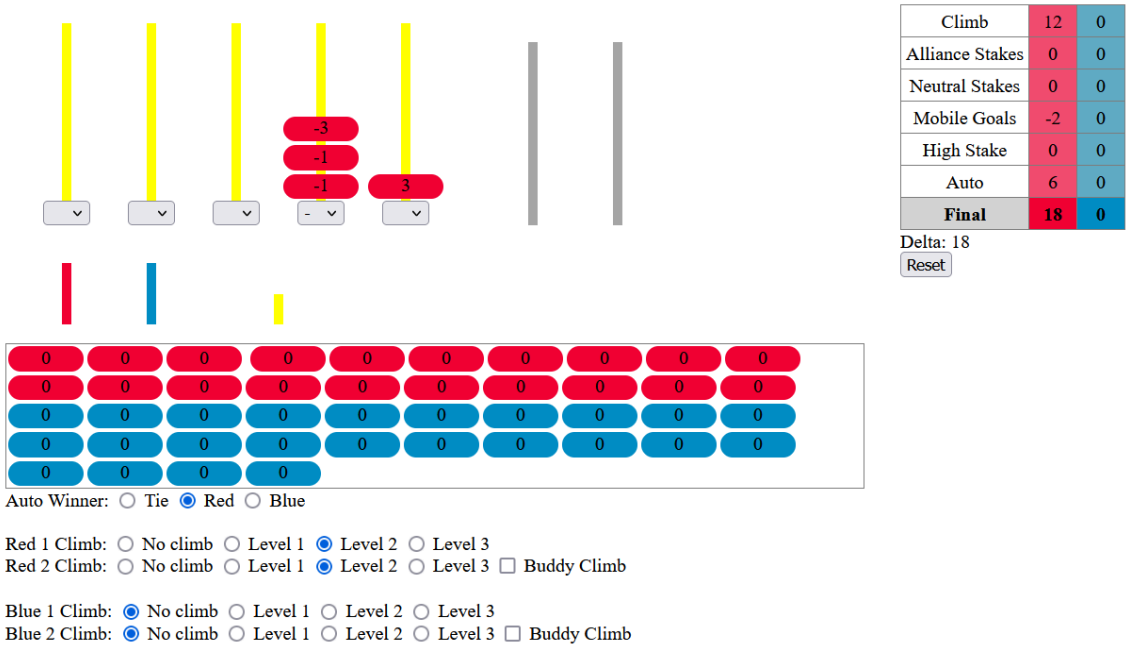
Another issue we found while testing the website is that the negative modifier took points away from the autonomous bonus.

<SC6biii> A Mobile Goal that has been Placed will result in the Corner modifiers to its Scored Rings

This negator [Negative Corner] only applies to an Alliance's "Ring points." Points received for Climbing and the Autonomous Bonus cannot be removed.

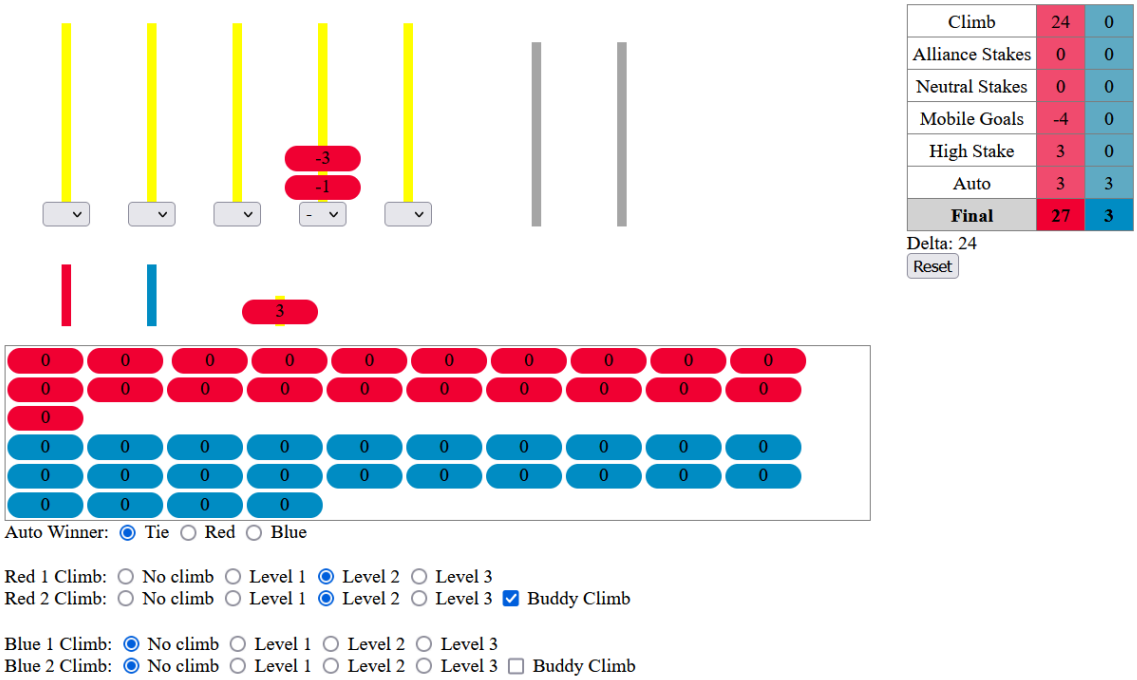


Incorrect Negative Modifier Scoring



Fixed Negative Modifier Scoring

Since the negative scoring modifier takes away from all stake points, it can currently incorrectly take away the High Stake Bonus. The High Stake Bonus is additional points for climb, however it is counted under high stake right now. This was fixed by adding the High Stake Bonus to the climb section instead of the high stake section.



Fixed Negative Modifier with High Stake Bonus Scoring

A skills section still has to be added, which will likely raise more issues than match scoring

Continued in **Skills Scoring Calculator** (Pg. 88)

External Notebook Generator Update 2

Focus: **Notebook Support Software**

By: **Richie Sommers** Date: **10/27/2024**

Continued from **Notebook Backup Fix** (Pg. 78)

Current Issues

The current generation script is functional but far from optimal. A lack of support for some non-standard but helpful markdown features, guides for reading the notebook, frontmatter, and a general ugliness means we still have work to do.

More Features

Goldmark, by default, does not support the following markdown extensions. However, we wish to use them for their benefits to understandability and for workflow improvements.

Highlighting

Use: Draws attention to specific, important values

Syntax: `==wow==`

Example: ==wow==

Inline & Block Comments

Use: Give comments when proofreading or as a place to store extra information that does not need to be in the final product.

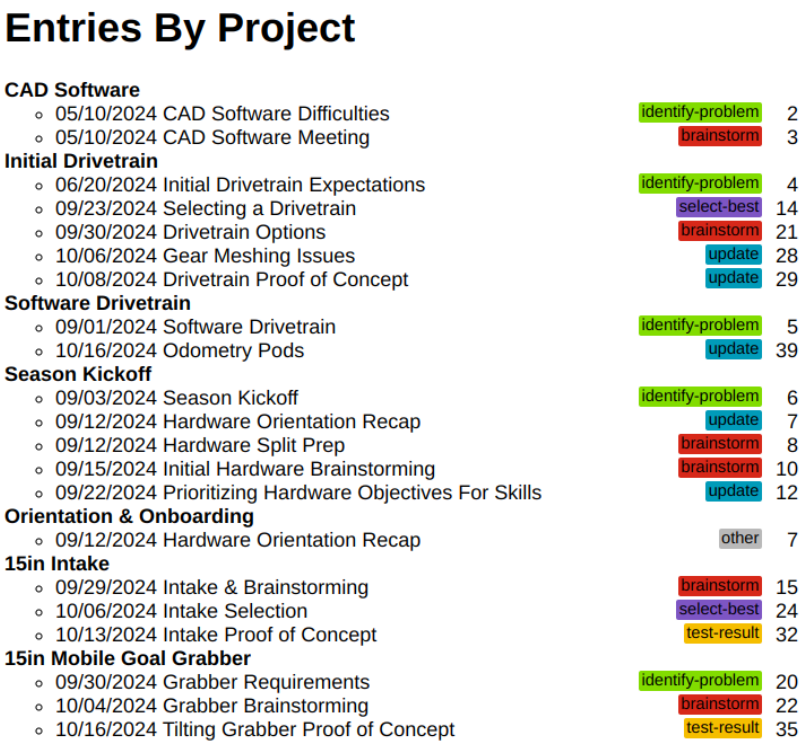
Syntax: `%comment text%`

```
%%
multiline
comment text
%%
```

Reading Guides

At this point, after further review of the notebooking guidelines, we are leaning towards putting entries in order of date. This provides a chronological view of all work over the course of the season telling the story of how designs grew and evolved.

However, this can lead to difficulty reading the final product if one only cares about a single project. An index showing all page numbers of a certain product is helpful, but it can be annoying to flip back to after reading each page.



Entries By Project

CAD Software
- 05/10/2024 CAD Software Difficulties — identify-problem 2
- 05/10/2024 CAD Software Meeting — brainstorm 3

Initial Drivetrain
- 06/20/2024 Initial Drivetrain Expectations — identify-problem 4
- 09/23/2024 Selecting a Drivetrain — select-best 14
- 09/30/2024 Drivetrain Options — brainstorm 21
- 10/06/2024 Gear Meshing Issues — update 28
- 10/08/2024 Drivetrain Proof of Concept — update 29

Software Drivetrain
- 09/01/2024 Software Drivetrain — identify-problem 5
- 10/16/2024 Odometry Pods — update 39

Season Kickoff
- 09/03/2024 Season Kickoff — identify-problem 6
- 09/12/2024 Hardware Orientation Recap — update 7
- 09/12/2024 Hardware Split Prep — brainstorm 8
- 09/15/2024 Initial Hardware Brainstorming — brainstorm 10
- 09/22/2024 Prioritizing Hardware Objectives For Skills — update 12

Orientation & Onboarding
- 09/12/2024 Hardware Orientation Recap — other 7

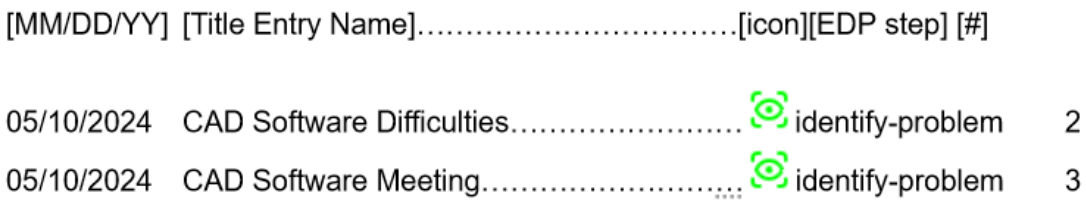
15in Intake
- 09/29/2024 Intake & Brainstorming — brainstorm 15
- 10/06/2024 Intake Selection — select-best 24
- 10/13/2024 Intake Proof of Concept — test-result 32

15in Mobile Goal Grabber
- 09/30/2024 Grabber Requirements — identify-problem 20
- 10/04/2024 Grabber Brainstorming — brainstorm 22
- 10/16/2024 Tilting Grabber Proof of Concept — test-result 35

Ugly Formatting

Many of the formatting decisions were already made while working on the internal notebook generator. However, newer additions such as the table of contents and page numbers need to have their styling improved.

Possible Table of Contents Styles



[MM/DD/YY] [Title Entry Name]……[icon][EDP step] [#]

05/10/2024 CAD Software Difficulties…… identify-problem 2

05/10/2024 CAD Software Meeting…… identify-problem 3

05/10/2024	CAD Software Difficulties		identify-problem	2
05/10/2024	CAD Software Meeting		identify-problem	3
06/20/2024	Initial Drivetrain Expectations		identify-problem	4

New Solutions

More Features

Highlighting

Highlighting syntax follows the same exact syntax as ~~strikethrough~~ (`~~strikethrough~~`) but with equals signs rather than tildes. Implementing highlighting uses the same parsing logic and renders with the built-in HTML `<mark>` tag.

Comments

#todo actually finish this

Reading Guides

To allow a reader to know the context of the project about which they are reading, we include a page number of the previous and next entries for a certain project. If the reader is trying to understand the full story of the team's work over the season, they simply go one page at a time. If the reader only cares about a single project, they can follow these page number guides.

Selecting a Drivetrain focus: **Initial Drivetrain**

by: **Mae Subramanian** reviewed by:
Continued from **Initial Drivetrain Expectations** (Pg. 4)

Continued in **Drivetrain Options** (Pg. 21)

Making it Prettier

Continued in **External Notebook Generator Update 3** (Pg. 87)

External Notebook Generator Update 3

Focus: **Notebook Support Software**

By: **Richie Sommers** Date: **10/27/2024**

Continued from **External Notebook Generator Update 2** (Pg. 84)

Automatic PDF creation turns wenapge that has some annoying requirements that the server is running in the right directory symbolic link Working directory containing css files with Assets/ from backup

assets folder is huge so better to not copy it but if we can make it happen in the filesytm then we can just use the builtin file server and not have to have funny handlers

go server could probably work can probably combine them, have go serve the files then cmd.Execute (node render.js)

%%https://stackoverflow.com/questions/35541589/creating-a-relative-symbolic-link-through-the-os-package%%

Skills Scoring Calculator

Focus: **Scoring Calculator**

By: **Zoe Rizzo** Date: **10/27/2024**

Continued from **Correct Match Scoring Calculator** (Pg. 80)

Now that match scoring is tentatively finished (we should still double check game manual rules and Q&As), a skills scoring section needs to be created.

Skills has a few main differences from matches:

- There are only 11 blue rings instead of the full 24
- Both blue and red rings can be scored on either alliance wall stake
- A mobile stake in a corner earns 5 points instead of a +/- modifier
  - Still only 1 mobile stake per corner
- Blue rings can only earn points if:
  - All red rings are scored and earn points
  - At least one red ring is below the blue ring(s)
  - There are no red rings scored above the blue rings
- If a red ring is scored above a blue ring, it is worth 0 points
  - This means that then all blue rings are worth 0 points

Notable game manual rules that may pose difficulty in programming:

<VURS4> Each blue Ring only has a point value if:

a. All red Rings in the Match have been Scored on Stakes and have point values b. At least one red Ring is Scored below the blue Ring(s) on that Stake c. No red Rings are Scored above the blue Ring(s) on that Stake

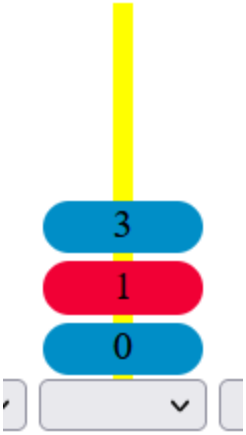
Note: <VURSC4> overrules the V5RC rule that states only one blue ring can be scored per stake.

<RSC5> Any red Ring Scored above a blue Ring on the same Stake will not have a point value.

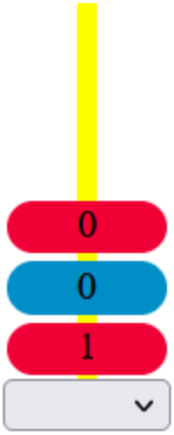
<RSC6> If any Ring is Scored on a Stake but does not have a point value based on rule <RSC4> or <RSC5>, no Ring on that Stake will earn points as a Top Ring.

The biggest difficulty of programming a skills scoring calculator will be keeping track of the rings positions-- currently, if a red ring is scored in between two blue rings, the red

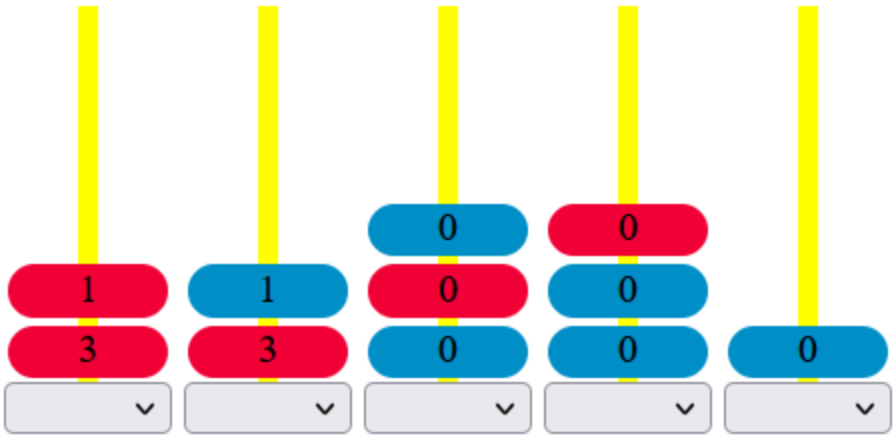
and blue rings are counted for points because the red position is not correctly tracked, just that there is a red ring below a blue ring.



In the above scenario, all rings should be worth 0 points: the bottom blue ring is not above any red rings and the red ring is scored above a blue ring. The top blue ring should be worth 0 since the red ring would be worth 0 points.

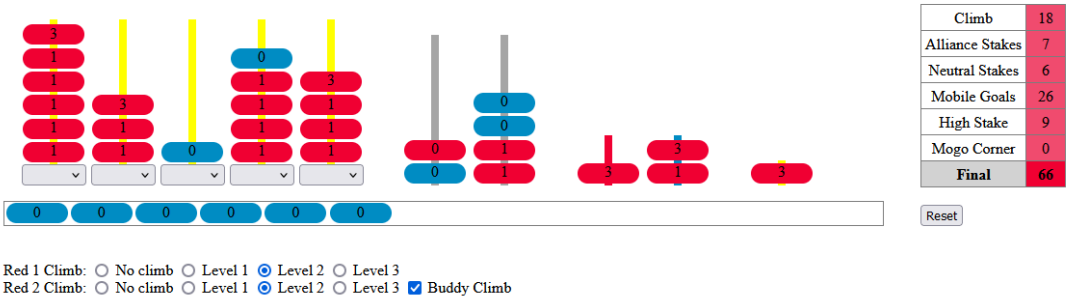


However, in this scenario, the bottom red ring would earn points as it does not violate any rules.

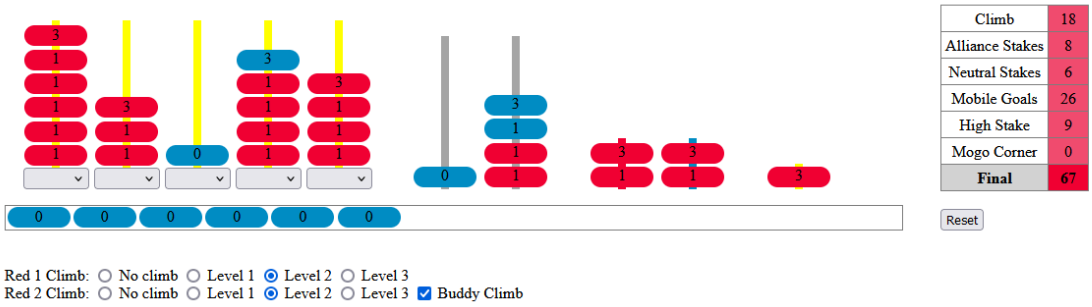


In the above scenario, all blue rings should be worth 0 points since some of the red rings do not earn points. The image above is also incorrect because the top ring should be worth 3 points, not the bottom ring. The bottom ring is currently worth 3 points because each stake is iterated over from bottom to top instead of top to bottom to check for bottom red rings. Additionally, not all red rings are scored, so blue rings should be worth 0 points.

To make sure all rings are scored correctly, all red rings are iterated through to make sure their score is not 0. Each stake is iterated over backwards to check to make sure there is a red ring on the bottom before any blue rings. Any blue rings are noted with a boolean value, and if any red rings appear after the boolean value is true, all blue rings and that red ring's scores are set to 0.

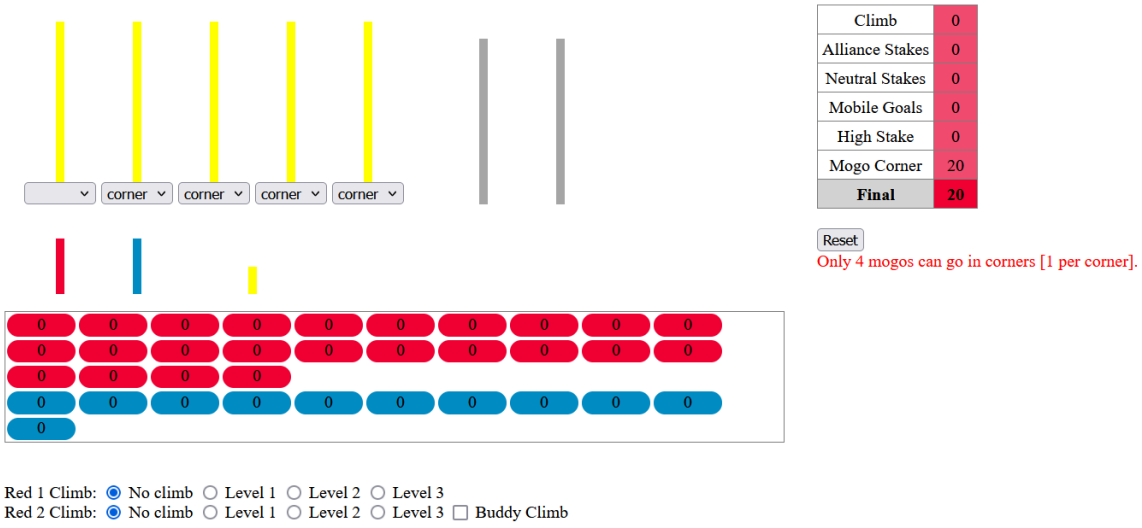


All Blue Rings Worth 0 Since Red Ring (Neutral Stake) Worth 0



Blue Rings Worth Points Since All Red Rings Scored and Worth Points

In addition to not allowing illegal actions, an error message is shown to tell the user why their action was not successful.



There are still a few changes to make that have been pointed out, both in match and skills scoring. In match scoring, we are currently scoring alliance wall stakes incorrectly. Also in match scoring, more than 2 mobile stakes can be placed in either +/- corner. Additionally, more error messages should be shown on both scoring pages to increase user experience.

Continued in **Scoring Calculator Errors** (Pg. 94)

Dead Wheels Update

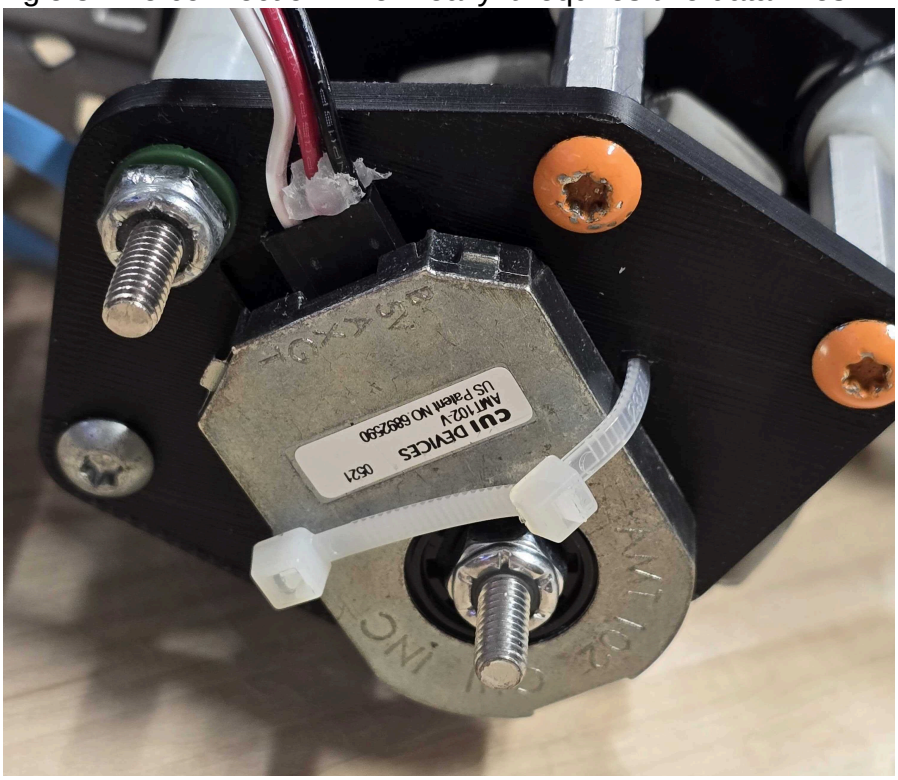
Focus: **Localization**

By: **Jack Cammarata**

Date: **10/28/2024**

Continued from **SparkFun Optical Tracking Odometry Sensor** (Pg. 77)

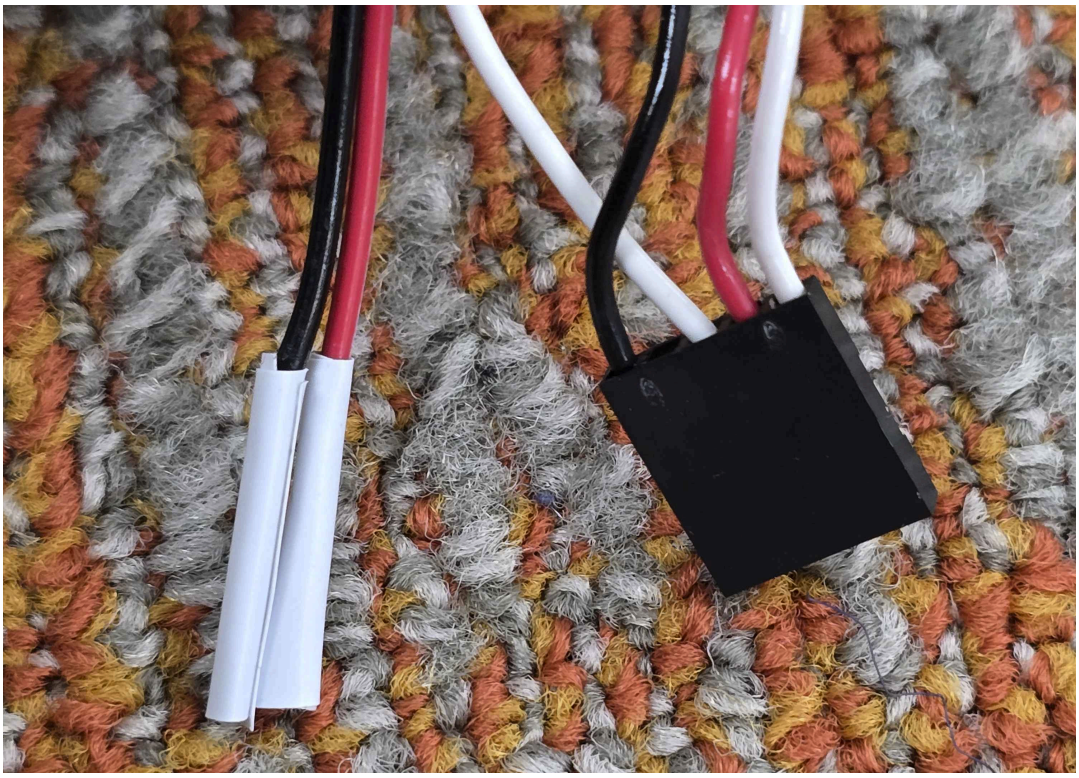
When the software drivetrain was built, the encoders not properly wired. Each one only had a single 3 wire connection when really it requires two data lines.



We only realized there was an issue because they were reading incorrect data, or rather no real data at all, because only one of the data lines was actually connected to the brain!

Wire	Encoder Connection
BLACK	Data B
RED	5V
WHITE	Data A
When using this setup, the encoder output `-0.043945` for both position and velocity regardless of movement, which is clearly incorrect.	

We can easily make a correct connector for these custom encoders by taking two standard 3-wire male to female cables, removing the female housings from both of them, connecting one set of wires to a 5 pin female housing, and only the white data line from the second cable. The remaining 5V and Ground wires from the second wire have their ends individually taped over to prevent accidental shorting.



Wire	Encoder Connection
WHITE1	Data B
RED1	5V
WHITE2	Data A
	X (unused)
BLACK1	Ground
With this setup, two 3-wire cables must be connected to the brain. In this case Triport A and Triport B. This now outputs correct position data according to the encoder's rotation.	

However, there are still some issues with the velocity measurement. Specifically, when moving very fast with high acceleration, the sign would sometimes be incorrect. Since our current implementation of both tank odometry and 3 wheel odometry both only use the wheel positions, this is not an issue. At some point we may want that functionality, but for right now we will ignore it.

By the end of this meeting, we created connectors for all three encoders on the software drivetrain and confirmed that they correctly track the wheel positions as the robot moves. More work still needs to go into an odometry implementation that can utilize 3 or more wheels.

Scoring Calculator Errors

Focus: **Scoring Calculator**

By: **Zoe Rizzo**

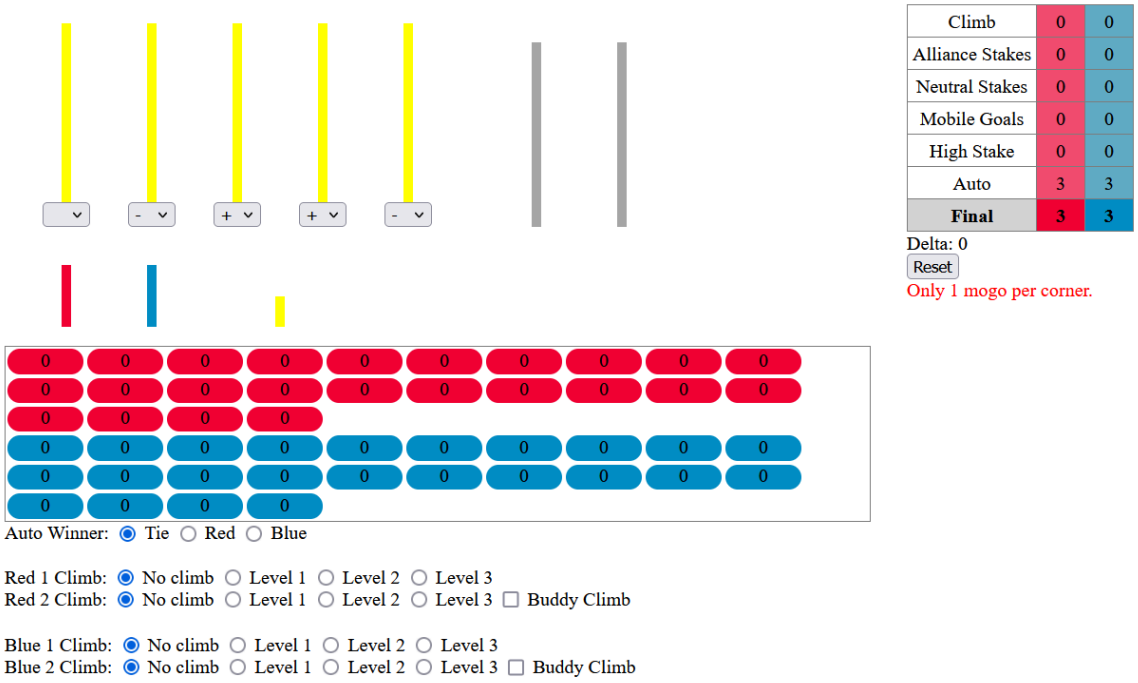
Date: **10/28/2024**

Continued from **Skills Scoring Calculator** (Pg. 88)

There are only two game manual inconsistencies left to fix with the scoring site: number of mobile stakes per corner and alliance wall stake scoring.

Mobile Goal Corner Modifiers

Per <SC5>, only one mobile stake may be considered placed in each corner. This means a maximum of two mobile stakes can have the + modifier, and a maximum of two mobile stakes can have the - modifier. Currently, the scoring calculator does not set a maximum for either modifier.



Fixed Mobile Goal Corner Modifier Limit

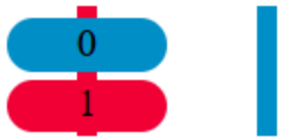
In addition to not allowing more than two + or - modifiers total, the user is shown an error message explaining why their action was not allowed.

Alliance Wall Stake Scoring

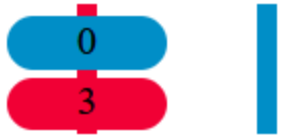
<Q&A 2060> If a Ring of the opposing color ends the Match in a Scored position on an Alliance Wall Stake, that Ring should not be considered as Scored, and will not earn points for either Alliance.

If it is the furthest Scored Ring from the Stake's base, the next-lowest Scored Ring on that Stake should be checked against rule <SC4>to see if it meets the criteria for Top Ring status.

Based on Q&A #2060, if the wrong color ring is scored as the top ring, and the correct color ring is scored below it, the correct color ring would be counted as a Top Ring.



The above scoring is incorrect, as the red ring should be scored as the Top Ring, earning 3 points.



The above scoring is correct since the red ring is scored as the Top Ring, earning 3 points.

Error Messages

While it is good to prevent users from performing illegal actions, it would also be good to show users why their actions could not happen.

Error Message	Description
Cannot add more than ${max_rings} rings to this stake.	Indicates to user that a stake cannot hold any more rings.
Only 1 mogo per corner.	Indicates to user that they could not successfully add a modifier to a mobile stake since the maximum number of modifiers has been reached.
Rings of the wrong color on alliance wall stakes earn 0 points.	Indicates to user that the ring just placed is on the incorrect alliance stake.
Red rings scored above blue rings earn 0 points.	Indicates to user that a red ring should not be placed above a blue ring.
Not all red rings scored. All blue rings worth 0 until all red rings scored.	Indicates to user that blue rings will not be scored until all red rings are scored.
Blue rings scored below red rings earn 0 points.	Indicates to user that a blue ring should not be placed below a red ring.
The current status of the scoring website is fully functional. All game manual rules are adhered to and the scoring calculator itself is fairly intuitive. The next steps will be to make the website mobile-friendly, as it was created and tested primarily on desktop, but would most likely be used more on mobile phones.	

Continued in **Mobile Friendly Scoring Calculator** (Pg. 97)

Mobile Friendly Scoring Calculator

Focus: **Scoring Calculator**

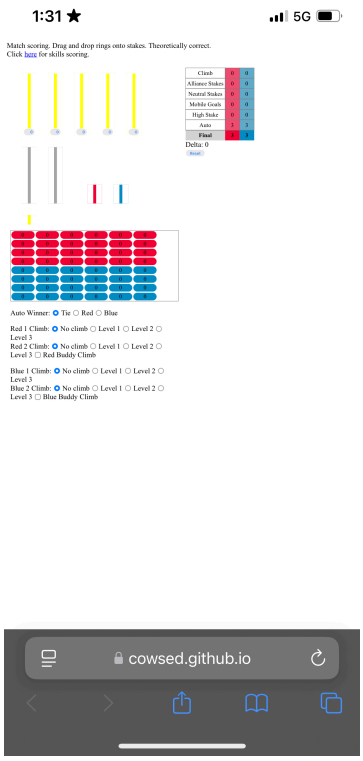
By: **Zoe Rizzo**

Date: **10/30/2024**

Continued from **Scoring Calculator Errors** (Pg. 94)

Now that the scoring calculator is compliant with all game manual rules and Q&As, it needs to be made mobile friendly. This is probably the most important part of the calculator, as it would be more ideal to use on a phone than a laptop.

Currently, the website is not mobile friendly at all.



The biggest issue is the way the HTML file is splitting the page: the scoring elements and score table are displayed together in a flexbox, each taking up 50% of the screen. While this is a good layout on desktop, it makes the calculator unviewable on mobile.

Instead of displaying the score table next to the scoring elements, the table should be displayed below them.

250	250

Since the score table cannot be viewed without scoring, the final score will be displayed at the top to users have quick access to the score.

The two other UI issues are with the high stake and climb. Each of those elements wrap around to the next row. The high stake should be pushed up to the same row as

the alliance stakes. To fix the climb elements, the best option is to turn the radios into selectors since a drop down menu will take up far less room.

300	300
Fixed Stakes Display and Score Displayed at Top	*Updated Drop Down Climb Scoring*
With these updates, the scoring calculator website is now completely accurate to the game manual rules and is user friendly enough to be considered finished. It should not have to be updated again unless there are changes to scoring in the game manual.	