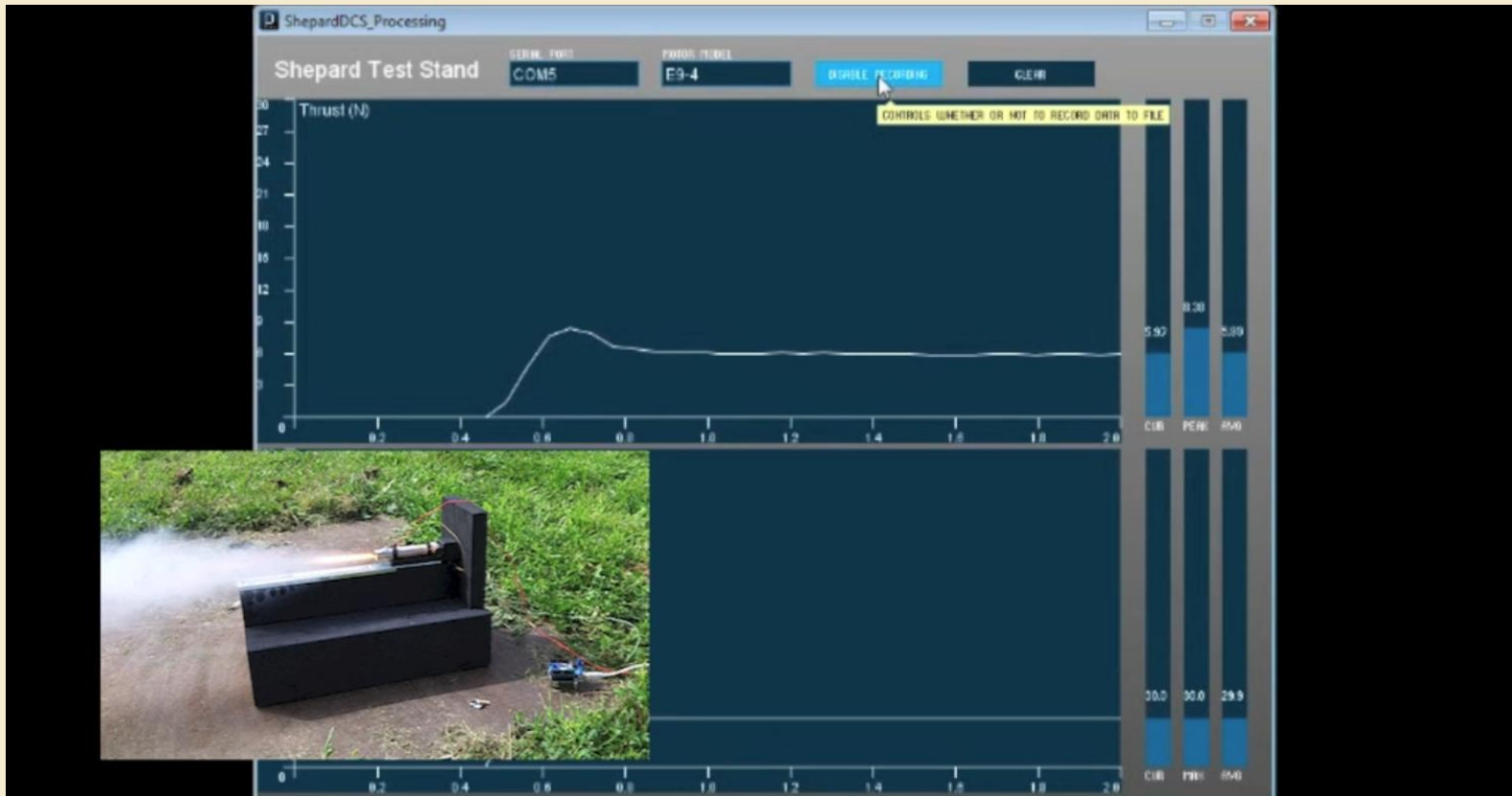


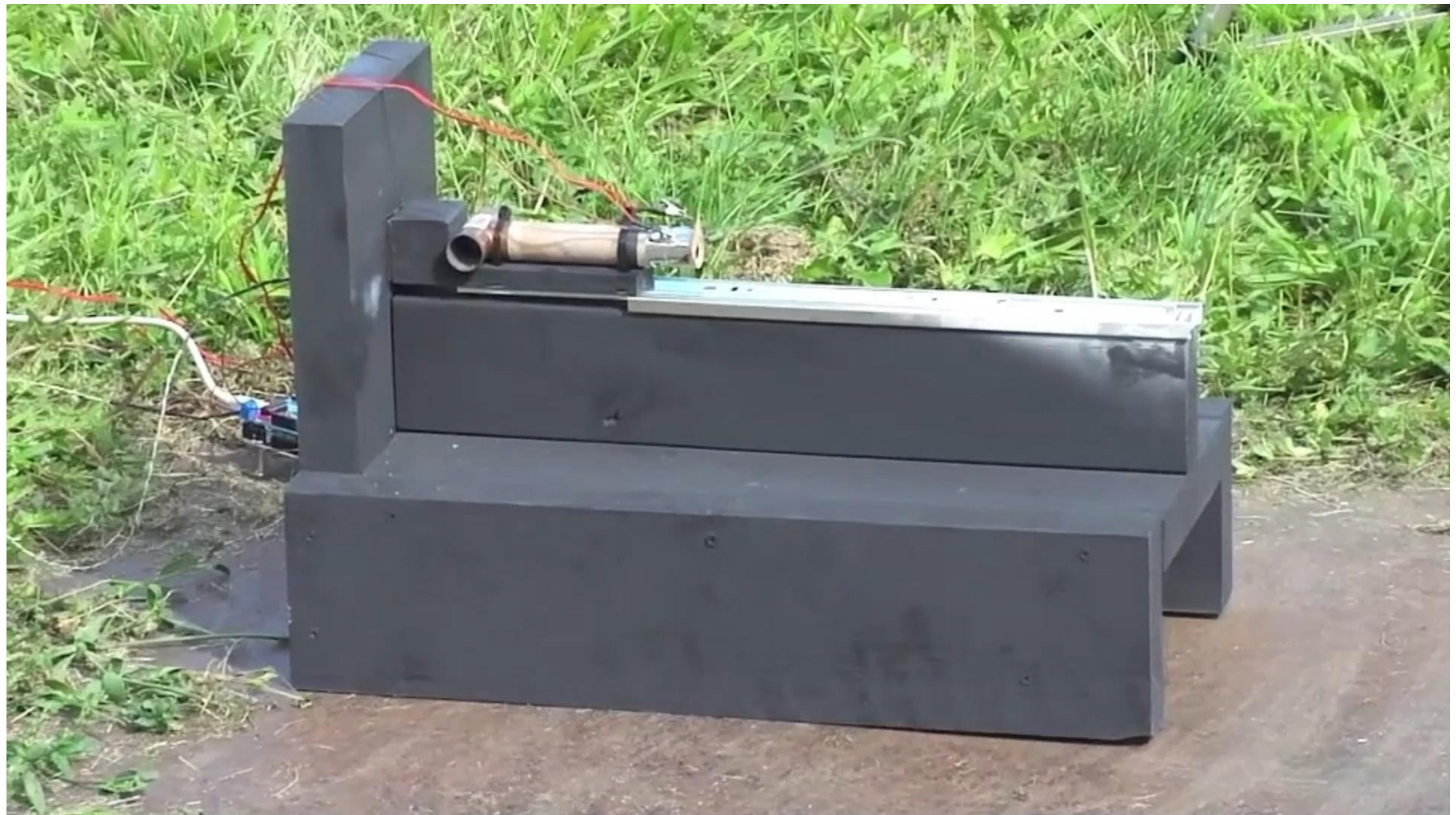
# Fire, Smoke, and Open Source Hardware

J. Simmons, Ph.D.  
thejsimmons.com



**J. Simmons, Ph.D.**  
Open Source Hardware Advocate





# Shepard Test Stand as OSHW Case Study

- Successes
- Lessons Learned
- Future Work

# J. Simmons, Ph.D.

## Open Source Hardware Advocate

### Founder of Mach 30: Foundation for Space Development

- US non-profit dedicated to developing OSHW for spaceflight
- Lead developer for the Shepard Test Stand ([OSHW Certificate US000006](#))
- Presented at the 2012, 2013, & 2015 Open Hardware Summits

### Chief Architect and Project Manager for Open Design Engine

- An all inclusive web application for hosting OSHW projects
- Kickstarted development of initial releast
- Currently focused on effective development, collaboration, sharing, and remixing of OSHW projects



# Developed Openly and Collaboratively on ODE

The screenshot displays the ODE project page for the Shepard Test Stand. The page includes a navigation bar with links for Home, Projects, and Help. The main content area is titled 'Shepard Test Stand' and features an 'Overview' section with a description of the test stand. A video player shows a 'Shepard Test Stand Demo Reel'. Below the video is an 'Issue tracking' section with a list of open issues. A large 'Members' box is overlaid on the page, listing the Manager, Developers, and Terms of Service Signers. The bottom of the page includes a footer with the Redmine logo and copyright information.

**Members**

Manager: 🌟 J. Simmons, 🌟 Jeremy Wright

Developer: 🌟 Aaron Harper, 🌟 Ben Barnett, 🌟 Christopher Sigman, 🌟 Ethan Chew, 🌟 Greg Moran, 🌟 Ryan Pulkrabek, 🌟 Wilton Burke

Terms of Service Signers: 🌟 J. Simmons, 🌟 Greg Moran, 🌟 Ben Barnett, 🌟 Jeremy Wright, 🌟 Wilton Burke, 🌟 Ethan Chew, 🌟 Aaron Harper, 🌟 Christopher Sigman, 🌟 Ryan Pulkrabek



# Developed Openly and Collaboratively on ODE

**Shepherd Test Stand**

Overview Activity Roadmap Issues Gantt News DMSF Wiki Forums Repository

### Roadmap

Bug  
 Feature  
 Support  
 Milestone  
 Show completed versions

v2.0 Shepherd Test Stand - Prototype  
 over 9 years late (07/30/2013)  
 Base for the kit version of the Shepherd Test Stand.  
 17%

v1.0 Shepherd Test Stand - Prototype  
 100%

v1.1 Shepherd Test Stand - Prototype  
 84%

**Related Issues**

- Bug #137: 64-bit Version of Java on Windows DoesNotSupport Serial Communications
- Bug #137: Time Stamp Not Working On Arduino and Client
- Bug #174: Create Java based GUI for Shepherd DCS
- Bug #175: Java GUI does not yet save CSV of data captured from the Arduino
- Feature #147: Add Mach 30 branding to Software Interface
- Feature #153: Switch to 5 Volt Version of MLX90614 IR Sensor
- Feature #154: Return 0.0 For Temperature if IR Sensor is Not Connected
- Feature #172: Add a "wizard" mode for the Shepherd GUI
- Feature #176: Add Compulsory Version Matching Between Arduino and Java App
- Feature #234: Arduino - Move Timestamp Zero Control to Arduino Instead of Client

**1.0 Shepherd Test Stand - Prototype**

This version of the test stand is a prototype to not only teach us about the fundamentals of building a rocket motor test stand, but also about the process of building spaceflight related hardware with a geographically distributed team. Many lessons were learned in this version that will be rolled into future versions of Shepherd and beyond.

**Related Issues**

- Milestone #010: Finish Shepherd v1.0 Documentation
- Milestone #010: Create Structural Assembly Instructions for Shepard 1.0

**v1.1 Shepherd Test Stand - Prototype**

**v1.1 Description**

The second iteration of the Shepherd rocket motor test stand. This is an incremental update over 1.0 where only the major issues are fixed, and the requirements are scaled back appropriately for the level of the project. There was a little bit of a process breakdown on this version, so some version 2.0 DAQ documentation got mixed in with the version 1.1 docs. Because of this, the wiki and dev logs may be a little hard to follow but have been cleaned up as much as possible. Please refer instead to the Shepherd v2.0 documentation to make sure that you are building the latest iteration.

**Shepherd Test Stand**

Overview Activity Roadmap Issues Gantt News DMSF Wiki Forums Repository

### Documents

Icon	Title	Size	Modified	Ver.	Author
[Data]	Data	516.9 KB	2012-09-11 12:09		Jeremy Wright
[Data]	Data Acquisition (DAQ) System Documents v1.x	9.9 KB	2012-12-21 09:48		Jeremy Wright
[Images]	Images	46.8 KB	2012-04-02 10:11		Jeremy Wright
[Manuals]	Manuals	400.4 KB	2012-09-24 09:45		Jeremy Wright
[Data]	Mechanical System Documents v1.x	3.4 KB	2012-09-21 12:03		J. Simmons
[Data]	Milestone Software Versions v1.x	3.1 KB	2012-09-21 12:09		Jeremy Wright
[Data]	Schematic and PCB Files v1.x	41.1 KB	2012-10-02 23:00		Jeremy Wright
[Data]	System Design Documents v1.x	279.8 KB	2012-09-21 12:09		J. Simmons
[Data]	Test Software v1.x	19.3 KB	2012-09-21 12:04		Jeremy Wright
[Data]	ShepherdTS Bolt v1.1	21.6 KB	2012-12-24 13:06	0.3	Jeremy Wright
[Data]	ShepherdTS Bolt v1_0	25.5 KB	2012-02-02 16:30	0.3	J. Simmons

Powered by Redmine © 2006-2011 Jean-Pierre Lang  
Terms of Service

**Shepherd Test Stand Wiki v2.0**

Introduction

Welcome to the project Wiki for the Mach 30 **Shepherd Test Stand**. This test stand holds small model rocket motors fixed (clamped in place) during firings so that things like thrust and casing temperature can be measured. This wiki contains documentation covering the design, development, fabrication, and use of the test stand.

The **Shepherd Test Stand** is named after a star Shepard, America's first astronaut, as it is our first test stand. We are using an Arduino board to provide the physical interface between the data collecting computer and the required sensors. This project is the first in a series of projects to develop the required skills for the practice of safe rocket engine operation, and to develop the capability to measure and record data about a rocket engine's performance. The idea behind the **Shepherd Test Stand** project is to start small and simple, and then build on what is learned when moving to larger and more complex stands later on. This is very much in keeping with Mach 30's philosophy of starting liberally from the ground up to build the infrastructure required to facilitate safe, modern, reliable, and sustained access to space. The second iteration of Shepherd only measured the thrust and motor casing temperature of Estes model rocket motors sized A through E. This third iteration (v2.0) will step beyond that to offer higher resolution measurements and an easier to assemble structure, among other things. The Shepherd project is structured in a way that should allow anyone to build and improve upon the designs. Everyone from experienced space flight hardware designers to educators and students are encouraged to get involved with the project and share their knowledge and enthusiasm.

If you're interested in getting involved, please introduce yourself in the forums and let us know what your interests and specialties are. There is also the navigation bar at the right to help you browse the documentation to get up-to-speed.

**Shepherd for Teachers and Scout Leaders**

We are currently working with a career organization, the Coca-Cola Space Science Center to develop curriculum so that Shepherd can be used in middle school, high school, and college classrooms. In addition, we intend to have Shepherd Test Stand kits available for sale sometime in the first half of 2014. If you're interested in finding out more about Shepard in education, please contact Mach 30. If you're planning to build and operate a Shepherd Test Stand on your own, please start by reading the Safety Procedures.

**Shepherd for Makers**

Mach 30 believes that Makers and Makerspaces will be a large part of the new spacefaring workforce, partnering with us in our mission to hasten the advancement of humanity into a spacefaring civilization through sustainable leadership, open design practices, and a bias toward mature technology. If that sounds like something you want to be involved in, we encourage you to jump right in. If you want to start with our design rationale, take a look at the Systems Engineering Process section in the navigation bar at the right. If building, testing, and operating a test stand is more your speed, start with the Documentation section of the navigation bar. Always make sure to read the Safety Procedures first so that you can have the safest experience possible with Shepherd. Also, don't forget to register for an Open Design Engine account so you can start posting to the forums.

**Shepherd for Students**

If you're a student who's ready to start working with Shepherd, be sure to start with the Safety Procedures. Once you're up to speed on those, you'll want to take a look at the Operating Manual to learn how to operate the test stand. Any other documentation that you may need should be in the navigation bar at the right under Documentation; also, don't forget to register for an Open Design Engine account so you can start posting to the forums.

**Contributing Makerspaces**

A Makerspace located in Indianapolis, Indiana with space (and tools) to collaborate on projects, a training room, a 3D printing area, and much more. Club Cybera graciously hosted integration testing for versions 1.0 and 1.1 of the Shepherd test stand.  
Website: <http://www.clubcybera.org>

**Shepherd Test Stand**

Overview Activity Roadmap Issues Gantt News DMSF Wiki Forums Repository

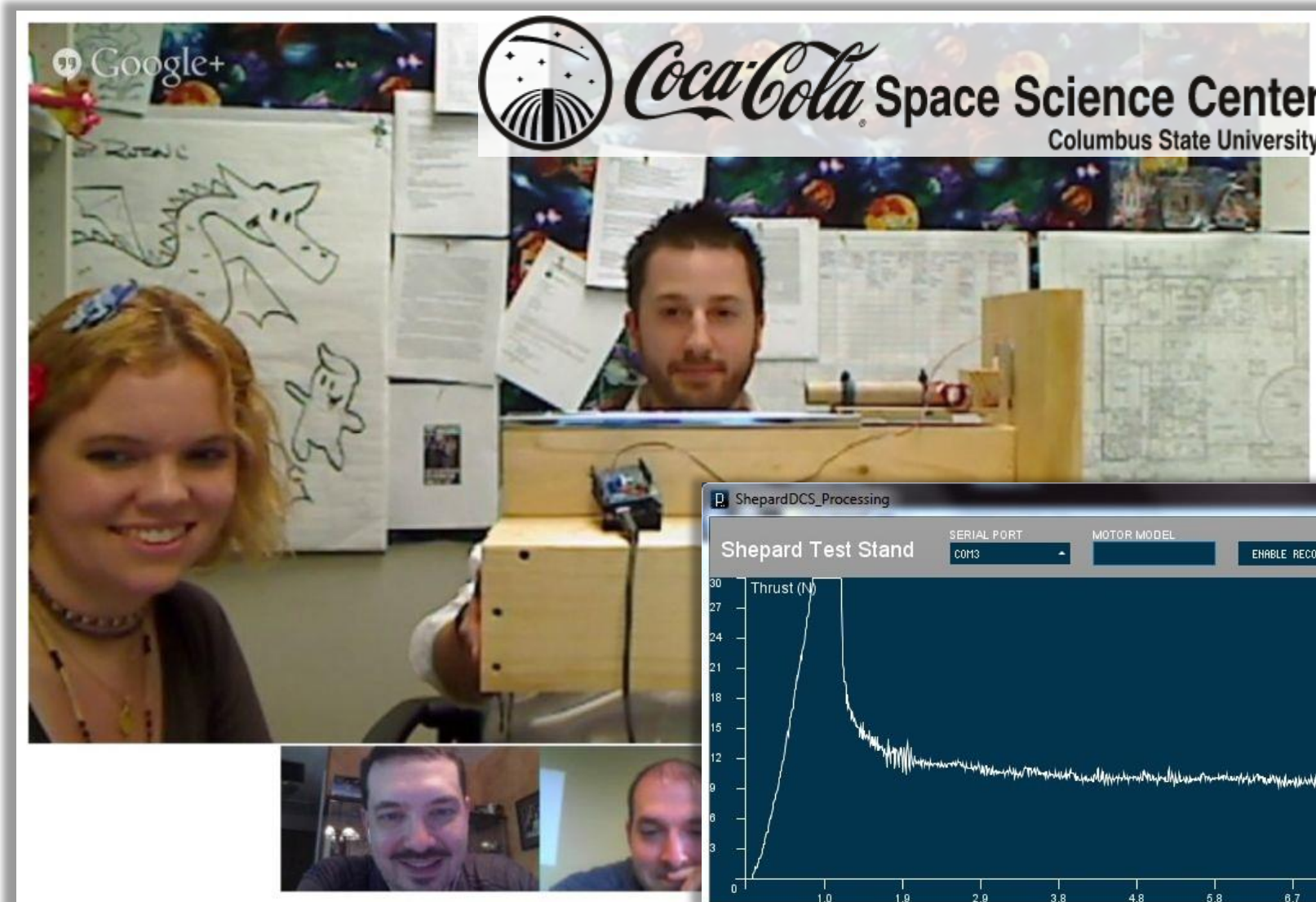
### Discussion

General engineering discussion of the Shepherd Test Stand

Subject	Author	Created	Replies	Last message
<b>Resources</b>				
Another desktop application environment to look	J. Simmons	08/02/2011 07:51 pm	53	Added by J. Simmons about 6 years ago
Socket For Molexix Temperature Sensors	J. Simmons	04/01/2014 11:50 am	9	Added by J. Simmons over 6 years ago
Jones Boys test fire	J. Simmons	05/07/2014 09:33 am	0	Added by J. Simmons over 6 years ago
Future Design Discussion - v2.0	J.C. Jones	12/08/2013 10:13 am	18	Added by J.C. Jones almost 9 years ago
Shepard Java Client	Jeremy Wright	01/29/2013 12:37 am	16	Added by J. Simmons over 9 years ago
Calibration and Modification	J.C. Jones	11/25/2013 09:33 pm	2	Added by J. Simmons about 9 years ago
v2.0 - SEP - Initial Questions	Jeremy Wright	09/23/2013 11:32 am	7	Added by J. Simmons about 9 years ago
v2.0 - SEP - Requirements	Jeremy Wright	10/09/2013 01:17 am	6	Added by J. Simmons over 9 years ago
Shepard v1.1 Wiki Docs Have Been "Tagged"	Jeremy Wright	09/30/2013 07:51 am	0	Added by J. Simmons over 9 years ago
Start Here Section of Shepherd Wiki	Jeremy Wright	09/18/2013 01:43 pm	17	Added by J. Simmons over 9 years ago
Dev Log Documentation	Jeremy Wright	09/23/2013 09:24 am	1	Added by J. Simmons over 9 years ago
Initial Questions	Jeremy Wright	09/23/2013 09:21 am	1	Added by J. Simmons over 9 years ago
Modification of Aaron's structure concept based on requirements dialog with CCSSC	J. Simmons	09/20/2013 11:35 am	4	Added by J. Simmons over 9 years ago
v1.1 Software interim download	J. Simmons	09/18/2013 08:40 am	0	Added by J. Simmons over 9 years ago
Testing updated Shepard DAQ code in preparation for Wed morning meeting with CCSSC	J. Simmons	09/17/2013 03:11 pm	6	Added by J. Simmons over 9 years ago
Shepard v1.1 vs Shepard v2.0 Process Breakdown	Jeremy Wright	09/14/2013 04:42 pm	0	Added by J. Simmons over 9 years ago
Dev Log Updates	Jeremy Wright	06/24/2013 09:15 am	10	Added by J. Simmons over 9 years ago
Accelerometer Uses on Test Stands	Jeremy Wright	09/10/2013 05:08 pm	0	Added by J. Simmons over 9 years ago
09-05-13 Shepard Hangout Follow-Up	Jeremy Wright	09/05/2013 10:29 pm	0	Added by J. Simmons over 9 years ago
v1.1 - Java Replacement for Processing Discussion	Jeremy Wright	08/27/2013 07:01 am	10	Added by J. Simmons over 9 years ago
v1.1 - STR 3.3 Issue	Jeremy Wright	08/27/2013 07:30 am	0	Added by J. Simmons over 9 years ago
v1.1 - SEP - Budget	Jeremy Wright	08/26/2013 07:26 am	0	Added by J. Simmons over 9 years ago



# Developed Openly and Collaboratively on ODE



# Developed Openly and Collaboratively on ODE



 **Coca-Cola** Space Science Center  
Columbus State University



# Received One of the First OSHWA OSHW Certifications

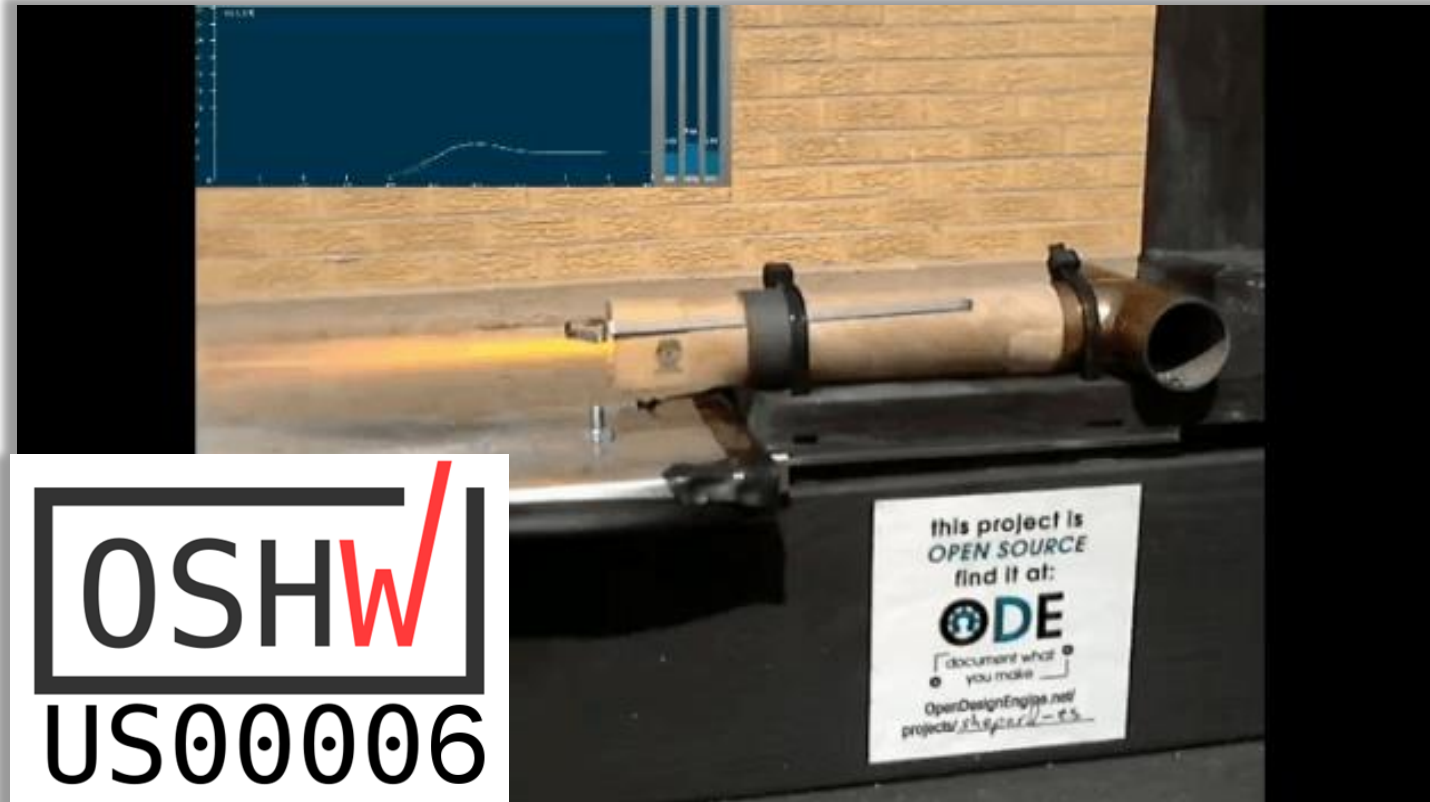
## OSHWA is a US Non-Profit



Photo credit OSHWA/OHS 2020

- Conferences and community events
- Educate general public about OSHW
- Organize the OSHW movement
- Collect, compile and publish data on OSHW movement
- Provide a painless way to indicate that products meet a standard for OSHW

# Received One of the First OSHWA OSHW Certifications



## MACH 30 SHEPARD TEST STAND

OSHWA UID  
**US000006**

PROJECT PROFILE

### TEST EQUIPMENT

The Mach 30 Shepard Test Stand is a test stand for Estes rocket motors. The Shepard Test Stand uses an Arduino Uno with a custom shield to provide the physical interface between the data collecting computer and the required sensors.

## Licenses

HARDWARE  
**OTHER**

The test stand is mostly functional. That means that it is unlikely that it is protected by copyright in a significant way. The Mach30 Open Design Pledge makes it clear to users that copyright restrictions will not prevent them from copying or improving upon the hardware.

SOFTWARE  
**APACHE**

The software related to the test stand is protected by copyright. The Apache license governs how others can copy and build upon the software.

DOCUMENTATION  
**CC BY**  
VIEW DOCUMENTATION

The documentation for the test stand is collected on the Mach 30 website. This documentation is protected by copyright and licensed under a CC-BY license. That means that others can copy and build upon the documentation as long as they give credit to Mach 30.

BRANDING  
INFORMATION

The Shepard Test Stand name mostly describes what the hardware is and what it does. Mach 30 might be able to obtain trademark protection for "Shepard" test stands if it wanted to use that name to differentiate test stands that come from Mach 30 from other test stands. The larger "Mach 30" name is unique and more than simply descriptive. Therefore Mach 30 could distribute their own Shepard Test Stands under the Mach 30 name and prevent others from using the same brand name on their products.

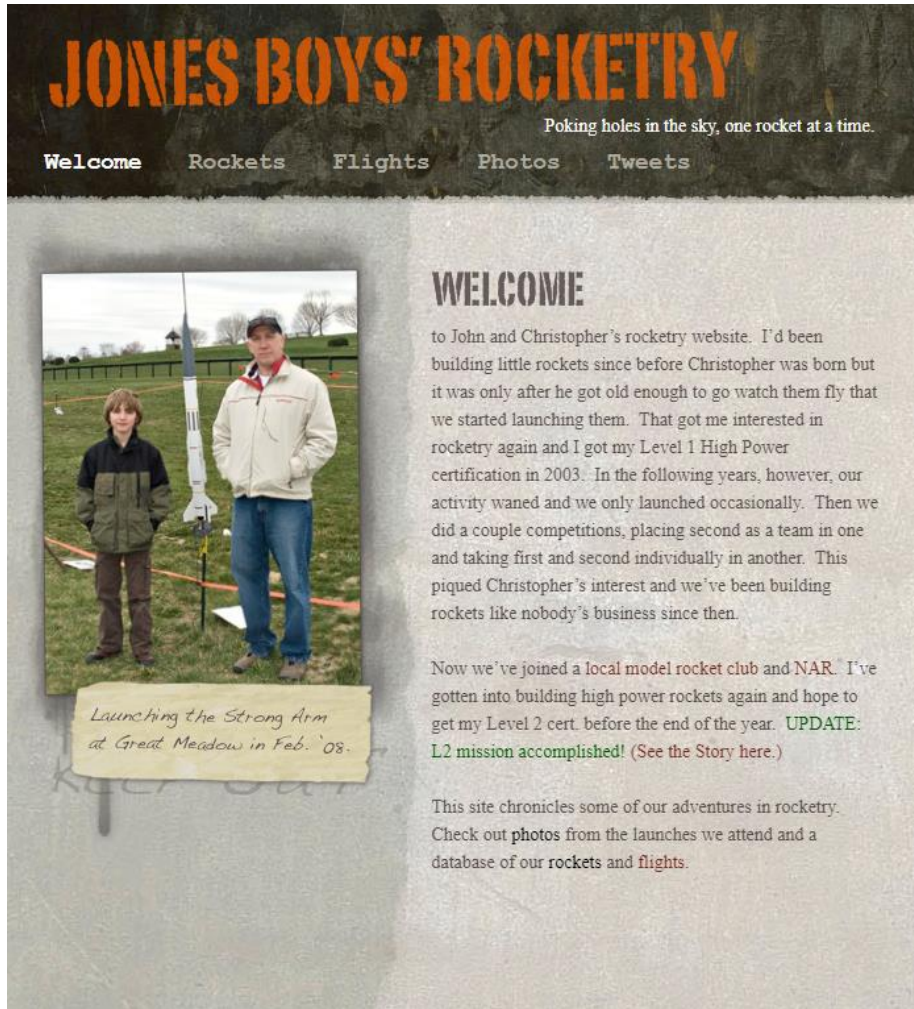
### PROTECTABILITY SPECTRUM OF PROJECT COMPONENTS

LESS  
PROTECTABLE

MORE  
PROTECTABLE



# Jones Boys Rocketry Forked Shepard



Shepard Test Stand Search:

Overview Activity Roadmap Issues Gantt News DMSF Wiki **Forums** Repository

Forums » Discussion »

## Calibration and Modification

Added by JC Jones about 9 years ago

Hello,

I'm helping my son with a science project on rocket motors. We have constructed a test stand and are trying to use your Arduino setup and software for the recording apparatus. We need to calibrate our load cell and it doesn't look like your Calibration Utility is working yet. How should we go about calibrating the setup so that we're getting accurate readings. Also, we also need to modify the scale of the measurements to accommodate higher thrust loads.

Thanks for your help,

John

### Replies (12)

RE: Calibration and Modification - Added by Jeremy Wright about 9 years ago

Hi John,

Glad to hear that you and your son are doing a rocket science project.

What version of the software are you running? The version 1.0 DAQ software is available in the [repository](#) if you haven't done so already.

Also, what version of the circuit did you build? There are known issues with the current version (and our latest circuit and procedures can be found in the [repository](#) with the Arduino Fritzing diagram). Make sure that you read the rest of the documentation to avoid the missteps that you'll want to avoid. The circuit outlined there will be your best bet. The calibration procedure that we're using right now is outlined as well. The calibration procedure is outlined in the [Estes motor documentation](#).

Please let me know if you need any other help.

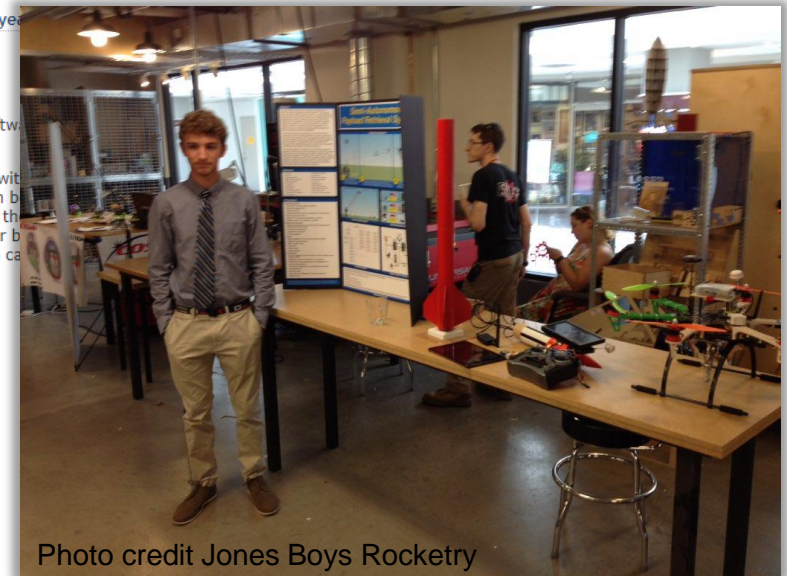
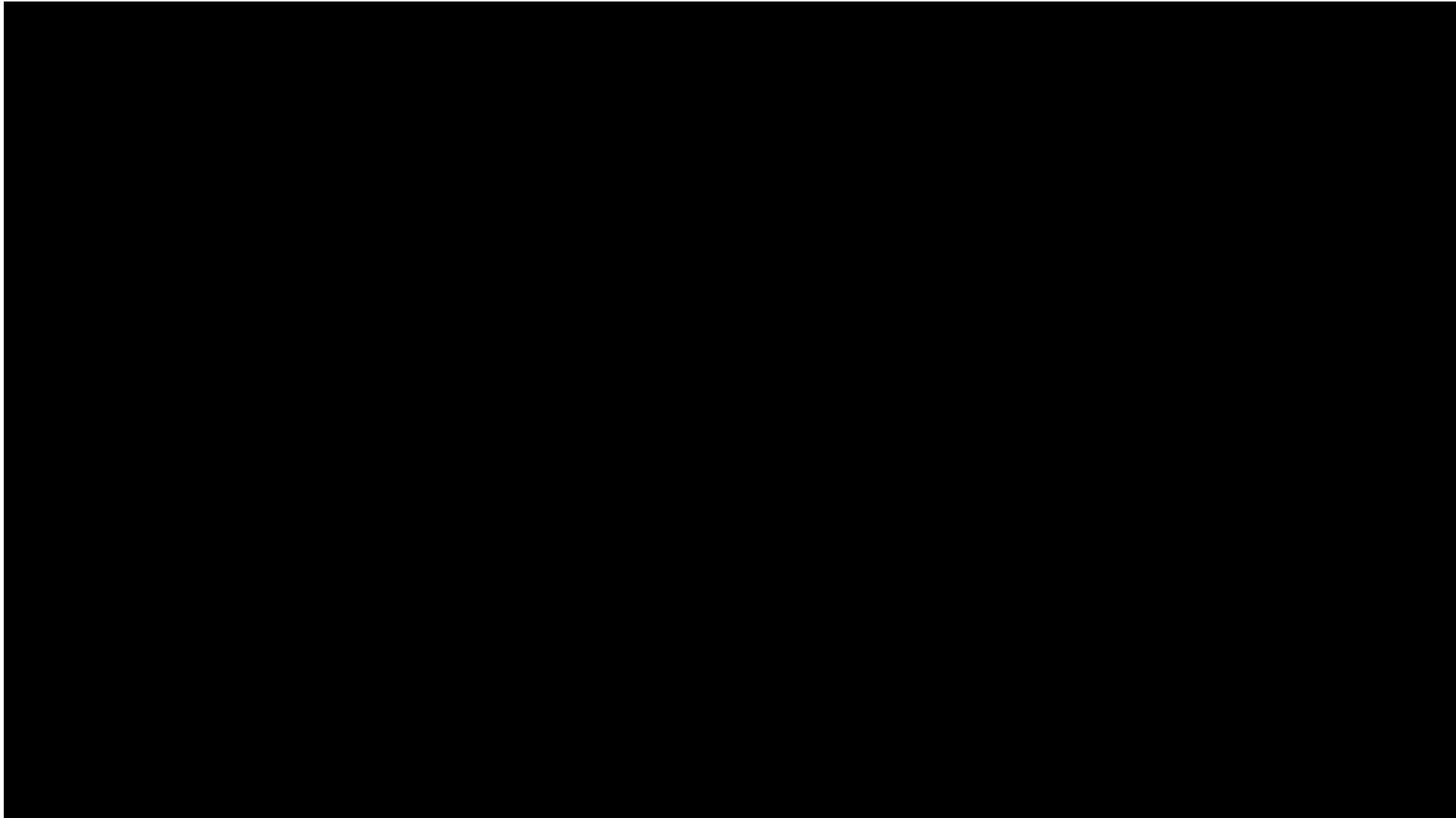


Photo credit Jones Boys Rocketry



# Jones Boys Rocketry Forked Shepard

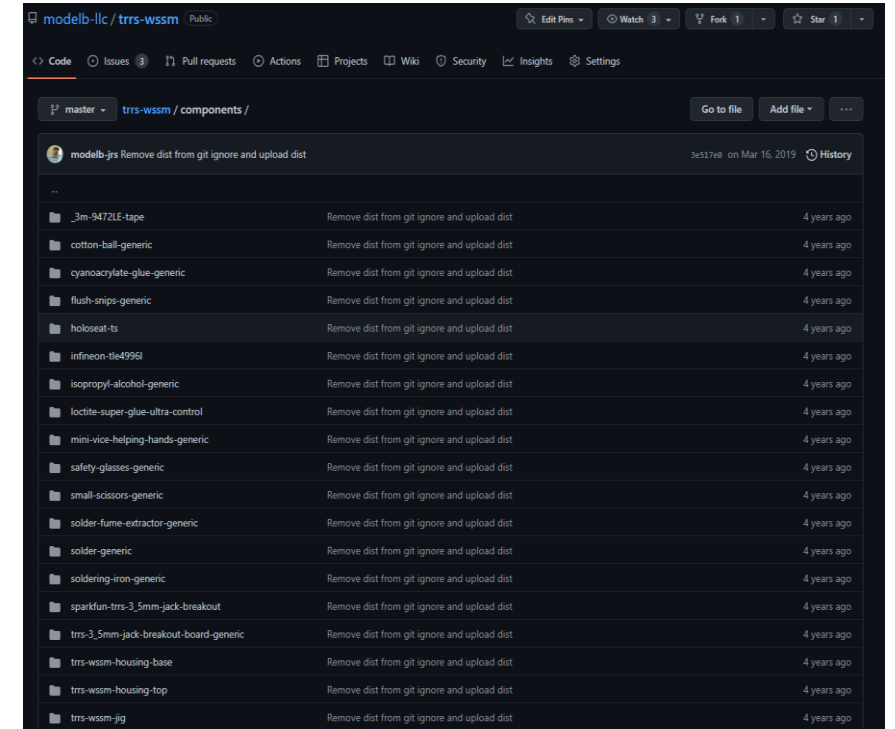


***“Open Source Hardware (OSHW) is a term for tangible artifacts — machines, devices, or other physical things — whose design has been released to the public in such a way that anyone can make, modify, distribute, and use those things.”***

from “[The Definition of OSHW](#)”



# open source hardware



## Table of Contents

*Distributed OSHW Framework (DOF)*

Introduction

The Pillars of the DOF

Methodology Documentation

Related Projects

Further Reading

Stakeholder Needs

User Stories

User Story 1: Branch OSHW

User Story 2: Fork OSHW

User Story 3: Merge OSHW

User Story 4: Composition of OSHW

Data Structures

Component

Component List Item

Activity Step

Parameter

Function

# Distributed OSHW Framework (DOF)

J. Simmons – [jrs@mach30.org](mailto:jrs@mach30.org) – Version V0.1.6, 8/6/2021

## Distributed OSHW Framework (DOF)

A. Simmons <[jrs@mach30.org](mailto:jrs@mach30.org)> :revnumber: v0.1.6 :revdate: 8/6/2021

### Introduction

Mach 30 launched [Open Design Engine](#) (ODE) in 2012. Since then we have run our own OSHW projects on ODE, observed other groups host OSHW projects on ODE and other sites, and held numerous conversations on and offline about the nature of hosting OSHW projects. Our conclusion after all these years and experience is the same one we held back in 2012: the OSHW community is still searching for a project hosting solution that meets the needs of hardware projects (where documentation is part of the source code).

What has changed is our approach to addressing OSHW project hosting. This time we are starting with the development of a tool independent methodology for developing and sharing OSHW, the Distributed OSHW Framework (DOF). What do we mean by methodology?

The Distributed OSHW Framework will be a systematic approach identifying:

- What needs to be shared
- How it should be shared
- The relationships between the various kinds of shared content

Note how developing a methodology is different from identifying best practices, covering case studies, and creating definitions. A methodology is something one follows: it is a fully defined process. And by targeting tool independence, we

methods have lasted through cvs,

<https://github.com/Mach30/dof>

# What is the source of an OSHW project?



1. Bill of Materials (BoM) Data ← **Keep the Project Source DRY**
2. Assembly Instruction Data ← **Humans are the Compiler**
3. Supporting Materials ← **Except when the 3D Printer is the Compiler**



# Branches, Forks, and Merges – Oh, My!



# It's Files All the Way Down

```

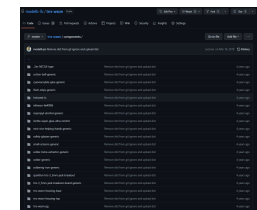
251 encoding_fallback_warning_issued = False
252 encoding_escape_warning_issued = False
253 def metadata_stream_to_writable_bytes(s):
254     encodingStrategy = gitConfig('git-p4.metadataDecodingStrategy') or defaultMetadataDecodingStrategy
255     fallbackEncoding = gitConfig('git-p4.metadataFallbackEncoding') or defaultFallbackMetadataEncoding
256     if not isinstance(s, bytes):
257         return s.encode('utf_8')
258     if encodingStrategy == 'passthrough':
259         return s
260     try:
261         s.decode('utf_8')
262         return s
263     except UnicodeDecodeError:
264         if encodingStrategy == 'fallback' and fallbackEncoding:
265             global encoding_fallback_warning_issued
266             global encoding_escape_warning_issued
267             try:
268                 if not encoding_fallback_warning_issued:
269                     print("\nCould not decode value as utf-8; using configured fallback
270 print("\n(this warning is only displayed once during an import)")
271 encoding_fallback_warning_issued = True
272 return s.decode(fallbackEncoding).encode('utf_8')
273 except Exception as exc:
274     if not encoding_escape_warning_issued:
275         print("\nCould not decode value with configured fallback encoding %s
276 print("\n(this warning is only displayed once during an import)")
277 encoding_escape_warning_issued = True
278 escaped_bytes = b''
279 # bytes and strings work very differently in python2 vs python3...
280 if str is bytes:
281     for byte in s:
282         byte_number = struct.unpack('>B', byte)[0]
283         if byte_number > 127:
284             escaped_bytes += b'%'
285             escaped_bytes += hex(byte_number)[2:].upper()
286         else:
287             escaped_bytes += byte
288     else:
289         for byte_number in s:
290             if byte_number > 127:
291                 escaped_bytes += b'%'
292                 escaped_bytes += hex(byte_number).upper().encode()[2:]
293             else:
294                 escaped_bytes += bytes([byte_number])
295 return escaped_bytes

```

```

869 903         if skip_info:
870 904             if 'code' in entry and entry['code'] == 'info':
871 905                 continue
872 -         if 'desc' in entry:
873 -             entry['desc'] =
874 -         if 'FullName' in ent
875 -         entry['FullName']
906 +         for key in p4KeysCon
907 +         if key in entry:
908 +             entry[key] =
876 909         if cb is not None:
877 910             cb(entry)

```

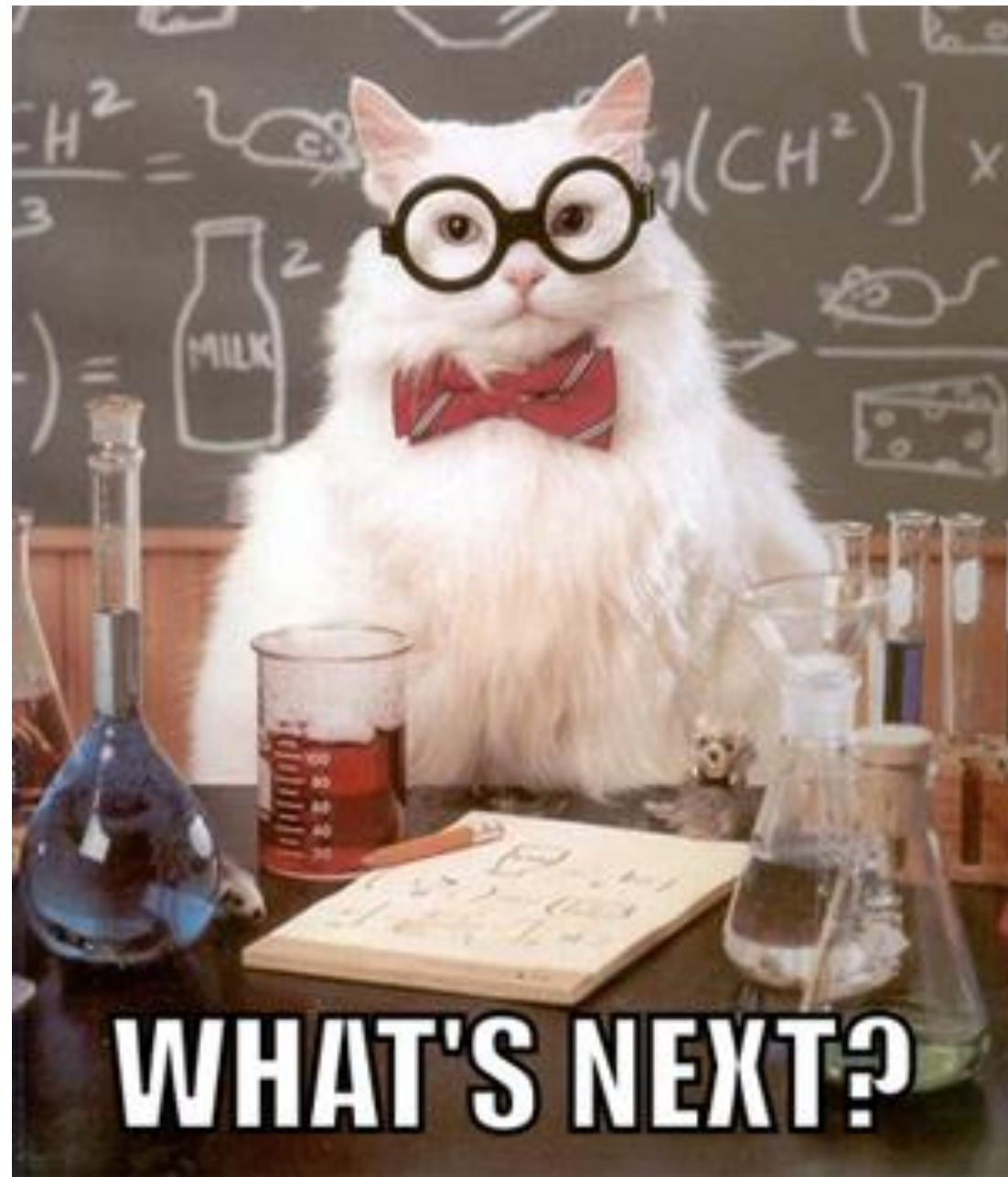


```

ndbroadbent@ndb-desktop ~/src/ubuntu_config [master±|1.9.2p290]$ gs
# On branch: master | +2 | [*] => $*
#
# Changes not staged for commit
#
#       deleted: [1]  _shared.sh
#       modified: [2] assets/git_breeze/config.example.sh
#       modified: [3] assets/git_breeze/config.sh
#       modified: [4] assets/git_breeze/git_breeze.sh
#       modified: [5] assets/git_breeze/lib/aliases_and_bindings.sh
#       deleted: [6] assets/git_breeze/lib/git_file_shortcuts.sh
#       modified: [7] assets/git_breeze/lib/git_repo_management.sh
#       deleted: [8] assets/git_breeze/test/git_file_shortcuts_test.sh
#       modified: [9] assets/git_breeze/test/git_repo_management_test.sh
#       modified: [10] assets/git_breeze/test/support/test_helper
#
# Untracked files
#
#       untracked: [11] assets/git_breeze/install.sh
#       untracked: [12] assets/git_breeze/lib/fallback/
#       untracked: [13] assets/git_breeze/lib/git_status_shortcuts.rb
#       untracked: [14] assets/git_breeze/lib/git_status_shortcuts.sh
#       untracked: [15] assets/git_breeze/test/git_status_shortcuts_test.sh
#
ndbroadbent@ndb-desktop ~/src/ubuntu_config [master±|1.9.2p290]$ |

```





# Defining DOF – Modeling OSHW

```

58 2-DataStructures/2-DataStructure.yaml: &ref_5
59   name: Data Structure
60   purpose: Define storage format for data used in an architecture
61   template: |
62     name: {{name}}
63     purpose: {{purpose}}
64     template: |
65       {{template}}
66     elements:
67       - {{element}}
68     derivedFrom: {{derivedFrom}}
69   elements:
70     - key: name
71       type: string
72       itemType: ''
73       description: Human readable name of the Data Structure
74       derivedFrom: []
75     - key: purpose
76       type: string
77       itemType: ''
78       description: Provides specification for an individual Data Structure
79       derivedFrom: []
80     - key: template
81       type: string
82       itemType: ''
83       description: >-
84         Liquid template of YAML representation of the data structure, suitable
85         for use in manual or automated generation of the data structure
86       derivedFrom: []
87     - key: elements
88       type: list
89       itemType: 2-DataStructures/1-Element.yaml
90       description: Definition of the data in the Data Structure
91       derivedFrom: []
92     - key: derivedFrom
93       type: list
94       itemType: string
95       description: >-
96         List of paths to model files that this Data Structure is related to by
97         analysis results and design decisions
98       derivedFrom: []
99     derivedFrom: []
100 2-DataStructures/3-Reference.yaml: &ref_6
101   name: Reference
102   purpose: Provide data required to cite sources

```



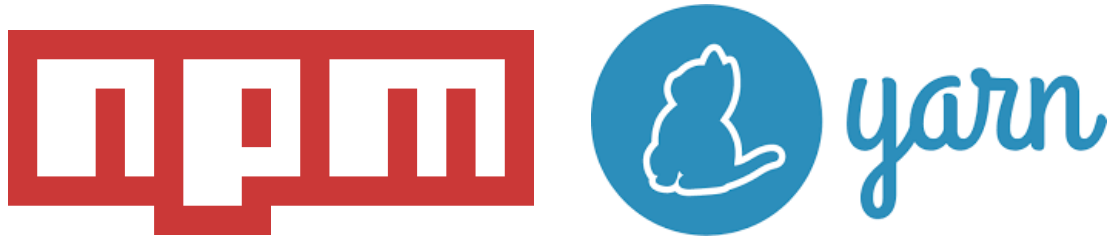
```

1  id: https://w3id.org/Mach30/m30ml/blob/main/src/linkml/root.yaml
2  name: m30ml_root
3  title: Mach 30 Modeling Language - Root schema
4  description: |-
5    Schema definitions for the Mach 30 Modeling Language (m30ml) Root Sy
6  license: https://creativecommons.org/licenses/by/4.0/
7  imports:
8    - linkml:types
9  prefixes:
10   linkml: https://w3id.org/linkml/
11   schema: http://schema.org/
12   m30ml_root: https://w3id.org/Mach30/m30ml/blob/main/src/linkml/root.yaml
13  default_prefix: m30ml_root
14  default_range: string
15
16  classes:
17    Element:
18      description: |-
19        The basic construct of an m30ml model. All other classes inherit from Element such that a model is a collection of Elements.
20        Elements may own other Elements via the *body* slot.
21      slots:
22        - id
23        - shortName
24        - name
25        - body
26
27  slots:
28    id:
29      identifier: true
30      required: true
31      description: |-
32        An element's [Globally Unique Identifier (GUID)](http://guid.one/guid) in a model.
33        Tools may choose the GUID version to use (language default is v4).
34      close_mappings:
35        - schema:identifier
36      # from https://www.geeksforgeeks.org/how-to-validate-guid-globally-unique-identifier-using-regular-expression/
37      pattern: "^[{}][0-9a-fA-F]{8}-([0-9a-fA-F]{4}-){3}[0-9a-fA-F]{12}[{}]*$"
38      shortName:
39        required: true
40        description: |-
41          An abbreviated name useful for referring to the element.
42          Conforms to the NPM package.json *name* [format](https://docs.npmjs.com/cli/v8/configuring-npm/package-json#name).
43      close_mappings:
44        - schema:name

```



# Defining DOF – Tooling & the Vision



```
npm install arduino-uno
```

**OR**

```
yarn add arduino-uno
```

## 2. Assemble TRRS Wheel Speed Sensor Module Housing Top

### 2.1. Tools

Name	Description	Notes
Personal Computer	Personal Computer (generic)	Windows, Mac, or Linux
Slicer	Slicer Software (generic)	User's choice of slicer software
3D Printer	3D Printer (generic)	Any FDM 3D printer compatible with selected filament

### 2.2. Materials

Quantity	ID	Name	Description	Notes
3.4 g	filament	Filament	PLA Filament (generic) v1.0.0	Thickness to match selected 3D printer specs

### 2.3. Procedure

#### Instructions

1. Launch the Slicer software on your Personal Computer
2. Open the *trrs-wssm-housing-top.stl* geometry file in the Slicer
3. (Optional) Suggested settings for Slicer, otherwise use default settings for selected Filament
  - A. Create Raft = True
  - B. Skirt Distance = Far (10mm)
  - C. Generate Support Material = True
  - D. Support Percent = 30%
4. Ensure Filament is loaded in the 3D Printer
5. Print the sliced geometry following the manufacturer's instructions for the 3D Printer
6. Remove the completed TRRS Wheel Speed Sensor Module Housing Top from the 3D Printer

<https://github.com/modelb-llc/trrs-wssm/blob/master/dist/assemblyInstructions.adoc>

# Supporting Aerospace @ OSHWA – m30ml

- DOF team is developing a base schema for DOF called m30ml
- m30ml supports capturing more generic concepts such as
  - Terms
  - Actors
  - Activities
- m30ml will enable the definition of open source standards to support certification of open source aerospace systems

# DOF Reference Tooling

## Existing Tools

- Git
- npm/yarn
- Rendering Layer
  - Pandoc
  - Jinja2
  - Markdown

## New Tooling Focus Areas

- Data Creation
- Data Maintenance
- Query/Content Extraction

aka CRUD operations



***“Open Source Hardware (OSHW) is a term for tangible artifacts — machines, devices, or other physical things — whose design has been released to the public in such a way that anyone can make, modify, distribute, and use those things.”***

from **[“The Definition of OSHW”](#)**





# What do you think?



**J. Simmons, Ph.D.**  
**thejsimmons.com**

