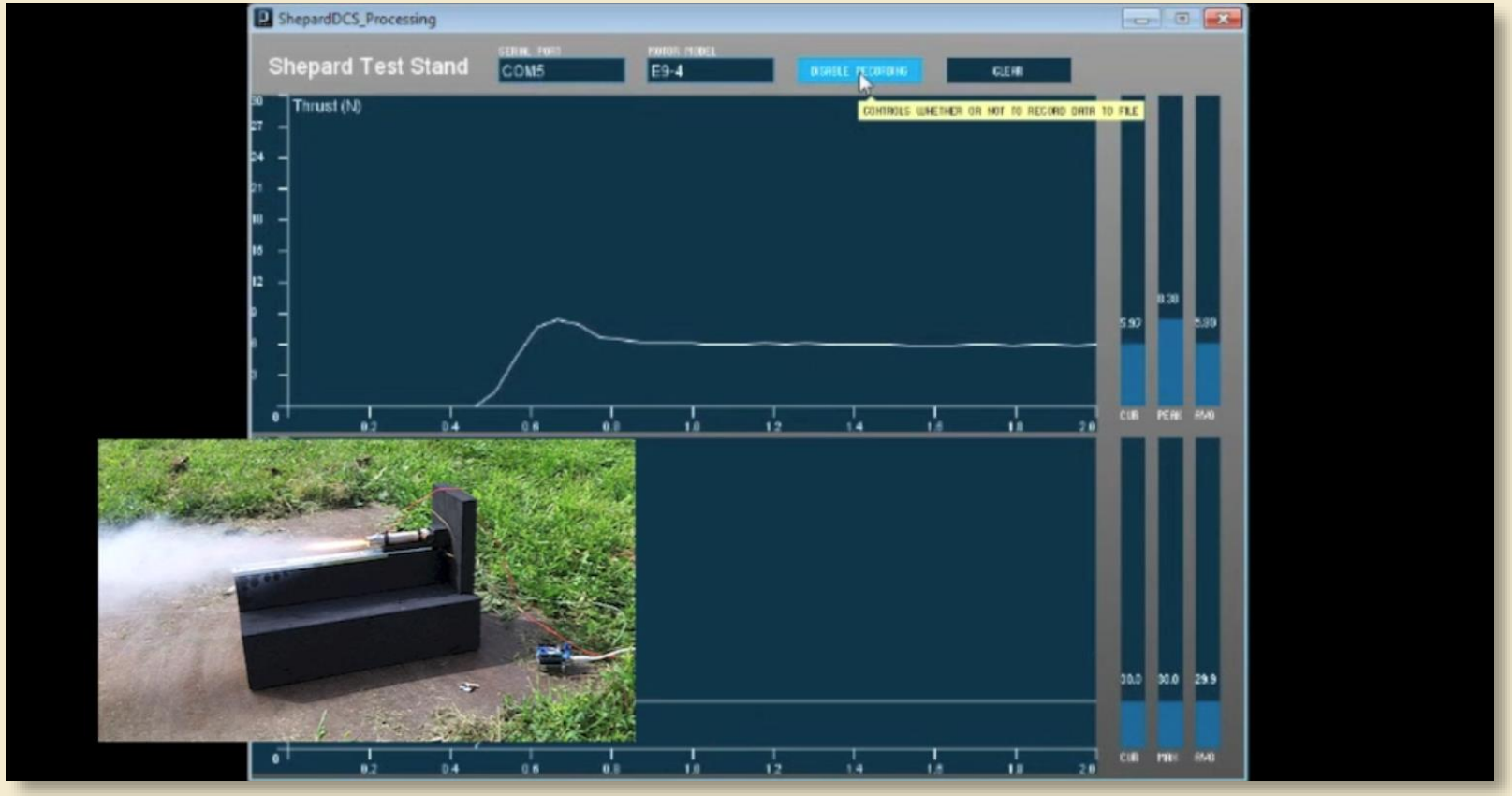
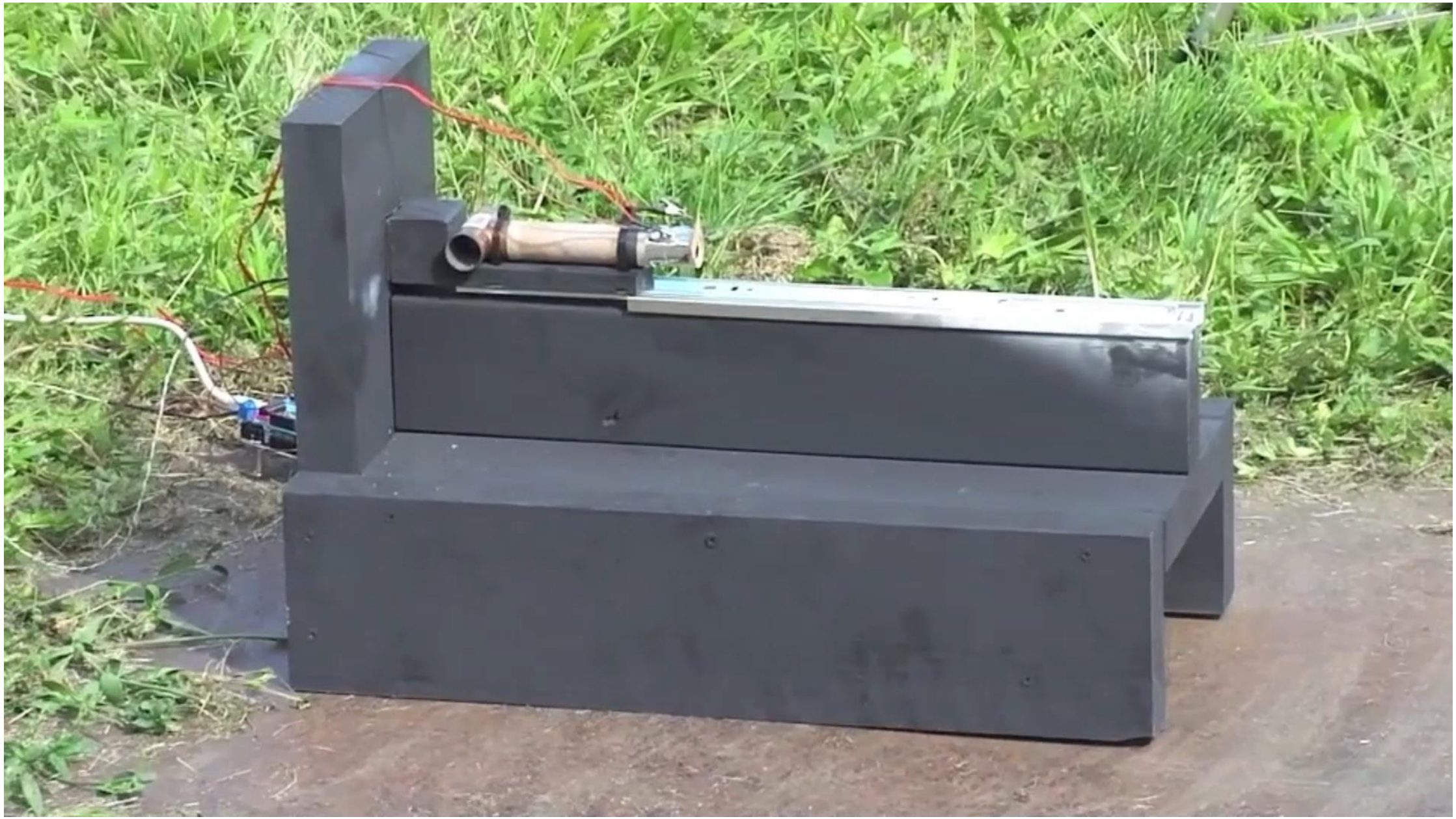
Fire, Smoke, and Open Source Hardware



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

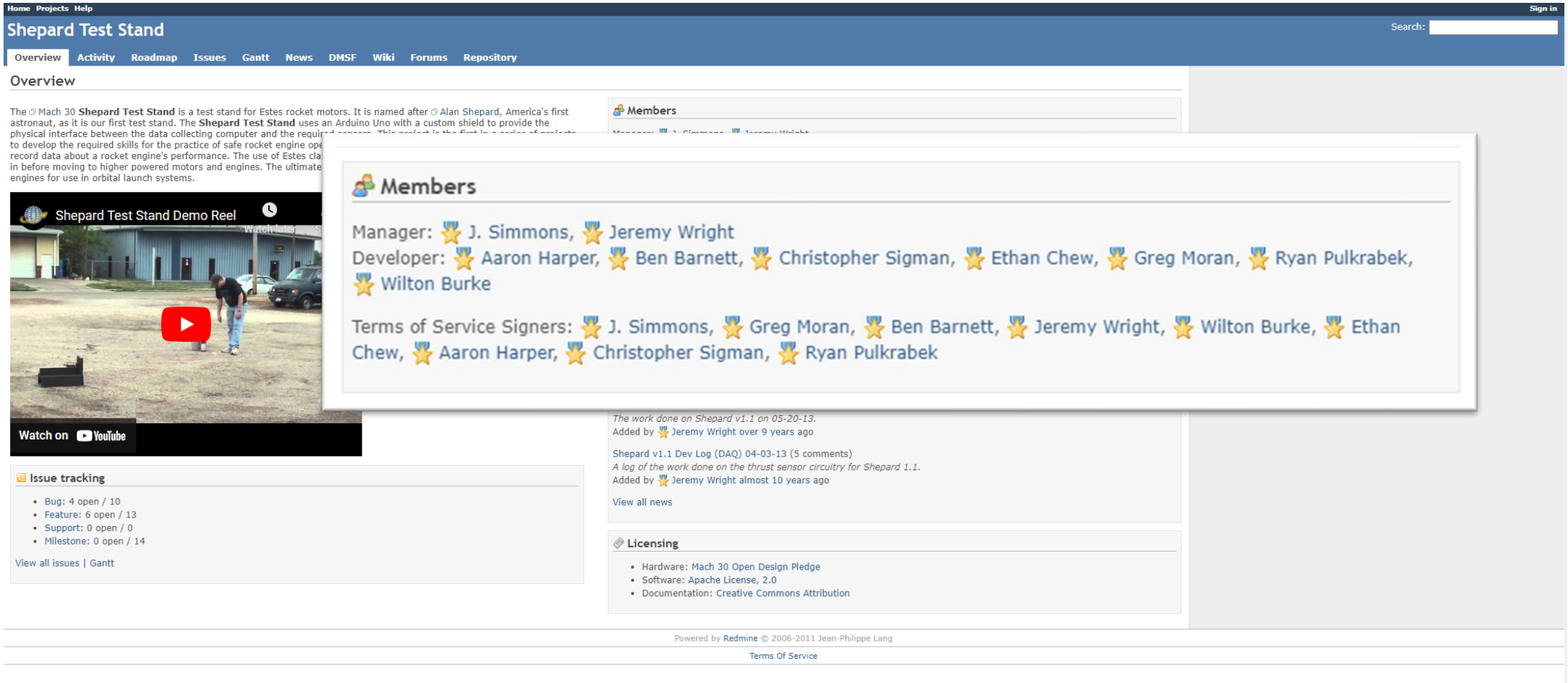
Founder of Mach 30: Foundation for Space Development

- 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

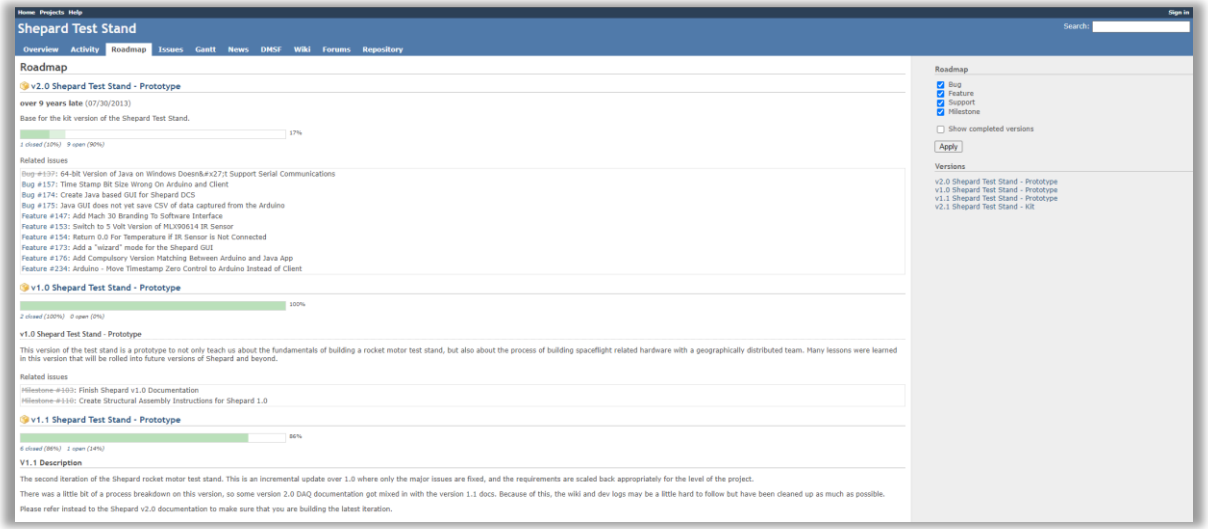
Chief Architect and Project Manager for Open Design Engine

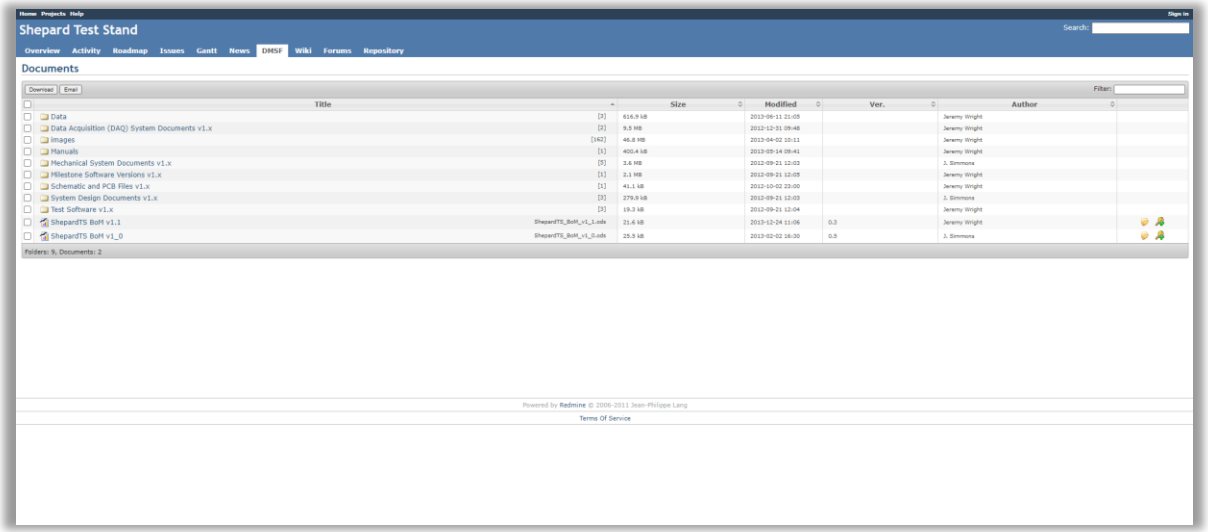
- An all inclusive web application for hosting OSHW projects
- Kickstarted development of initial releast
- Currently focused on effective development, collaboration, sharing, and remixing of OSHW projects

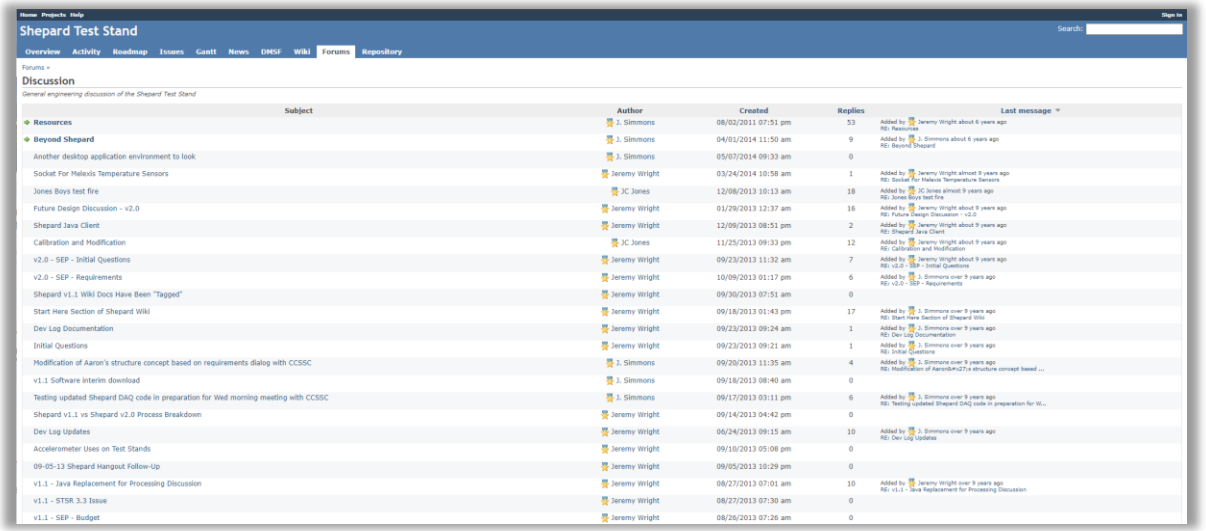
Developed Openly and Collaboratively on ODE



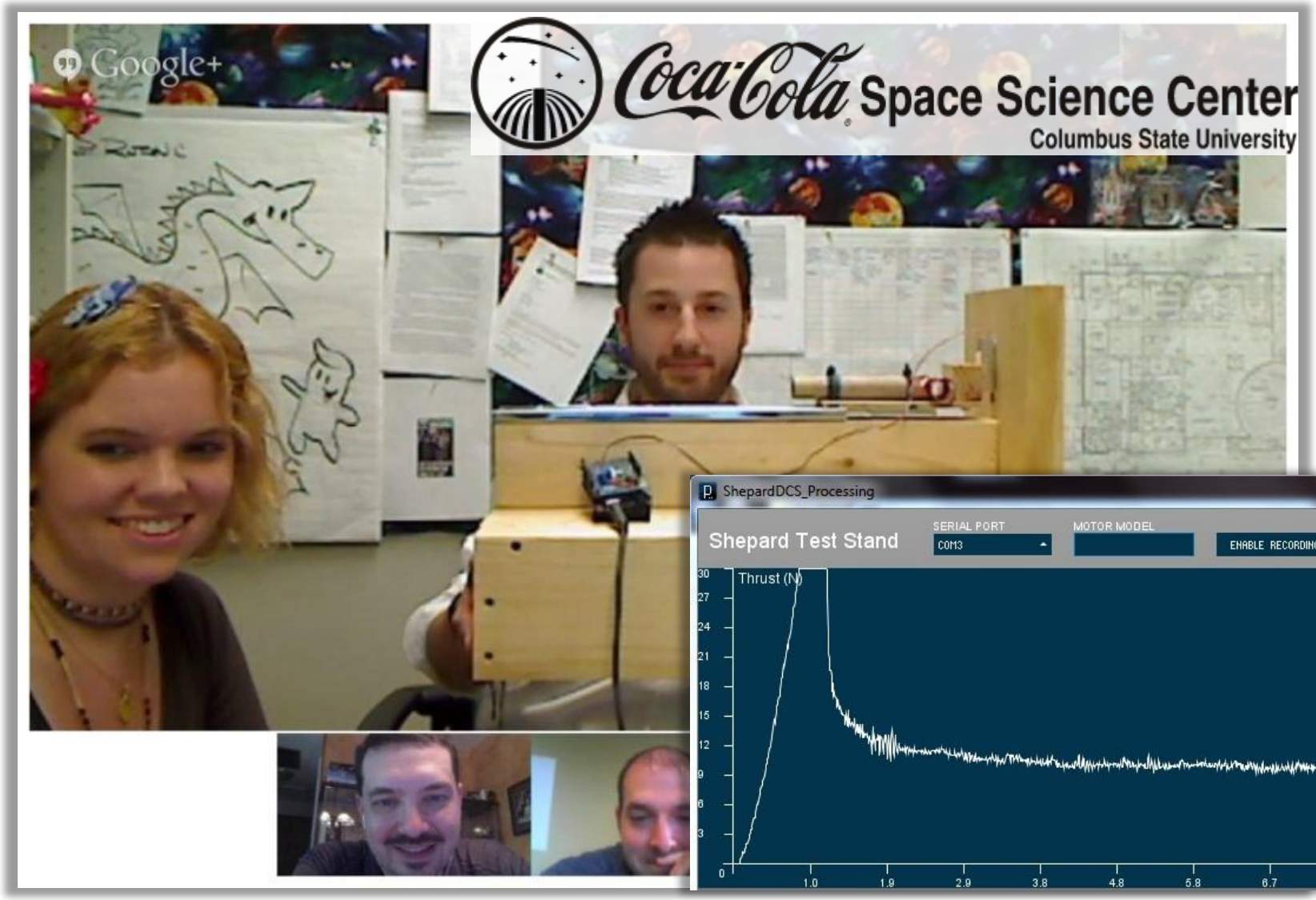
Developed Openly and Collaboratively on ODE







Developed Openly and Collaboratively on ODE



Developed Openly and Collaboratively on ODE

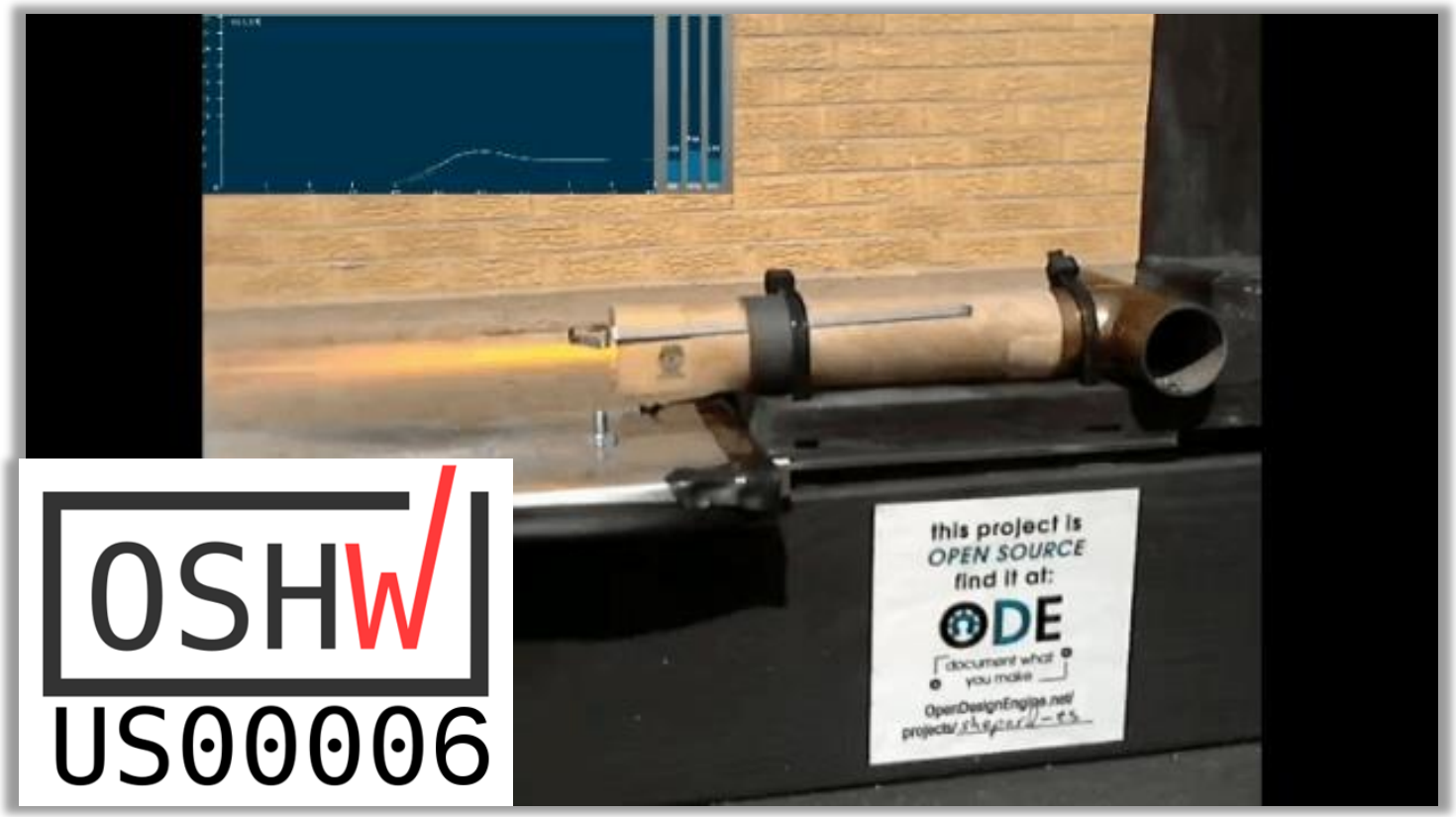
Received One of the First OSHWA OSHW Certifications

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.

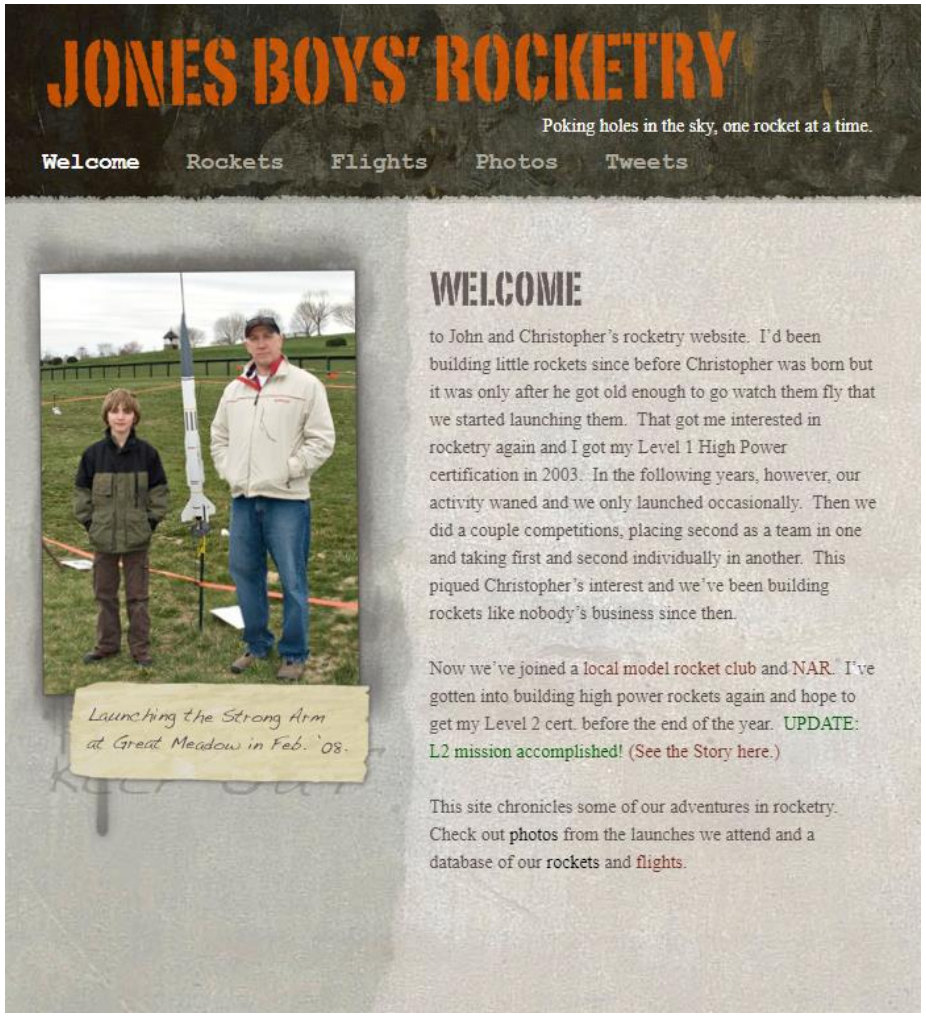
BRANDING INFORMATION

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



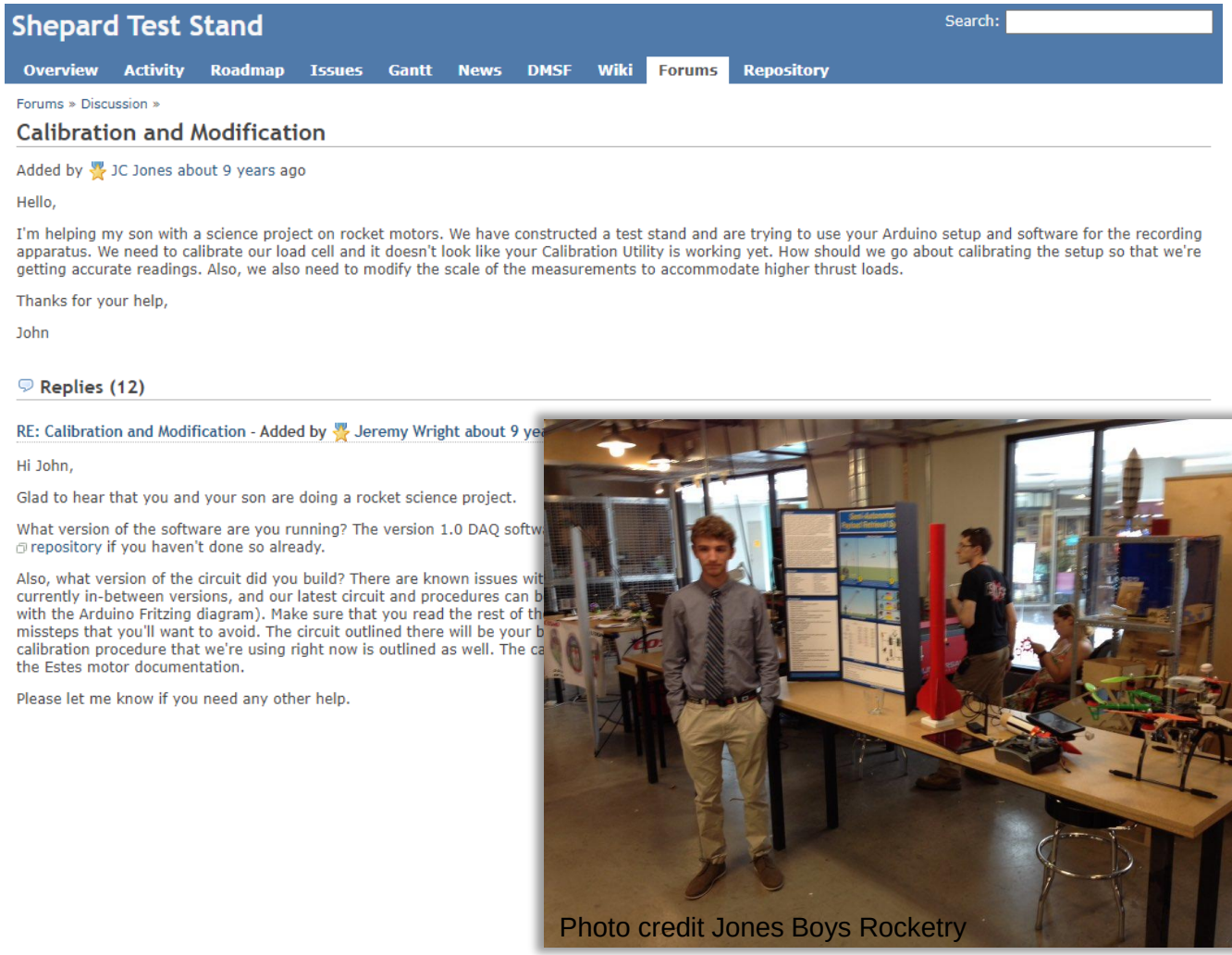
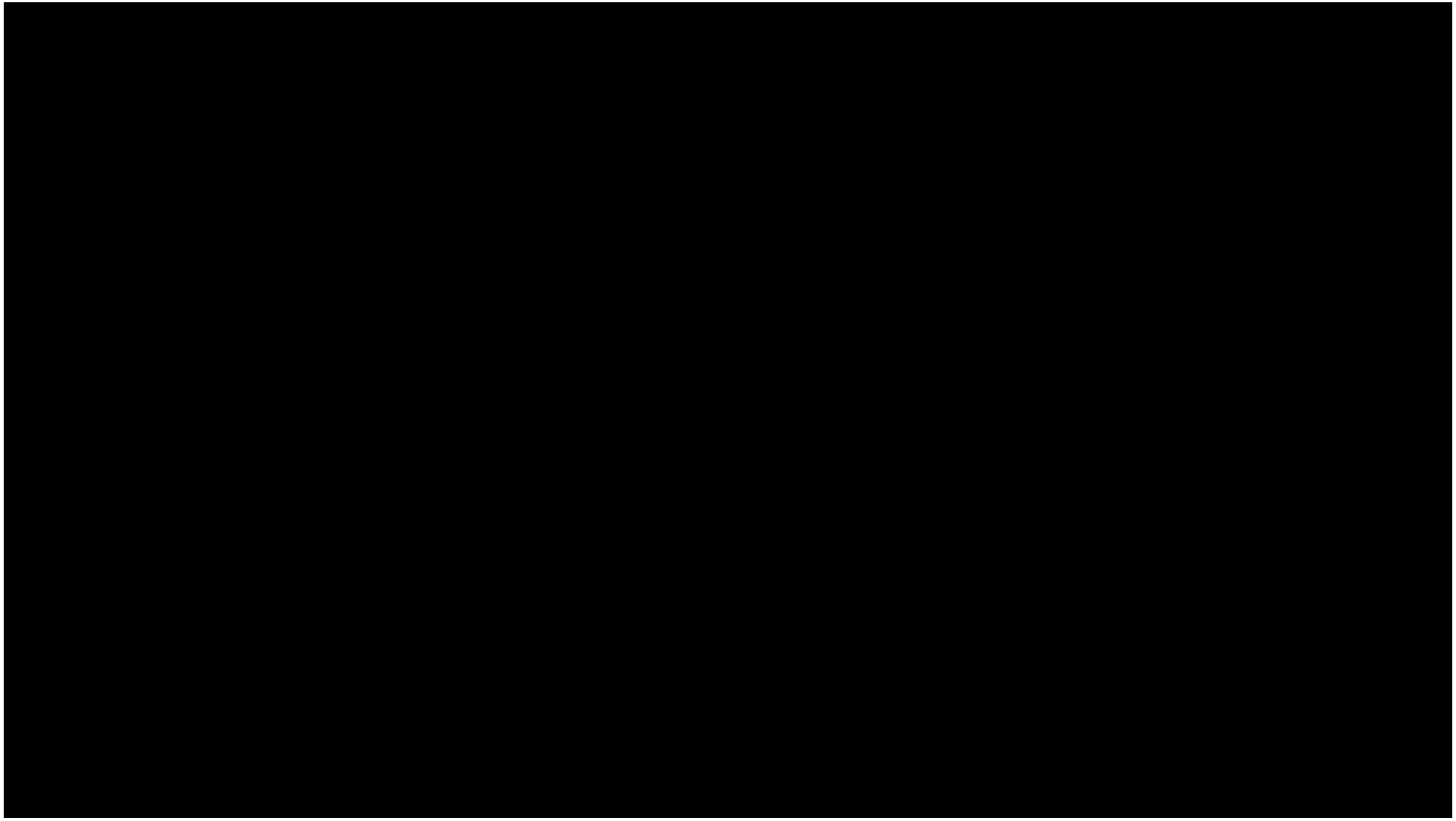


Photo credit Jones Boys Rocketry

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

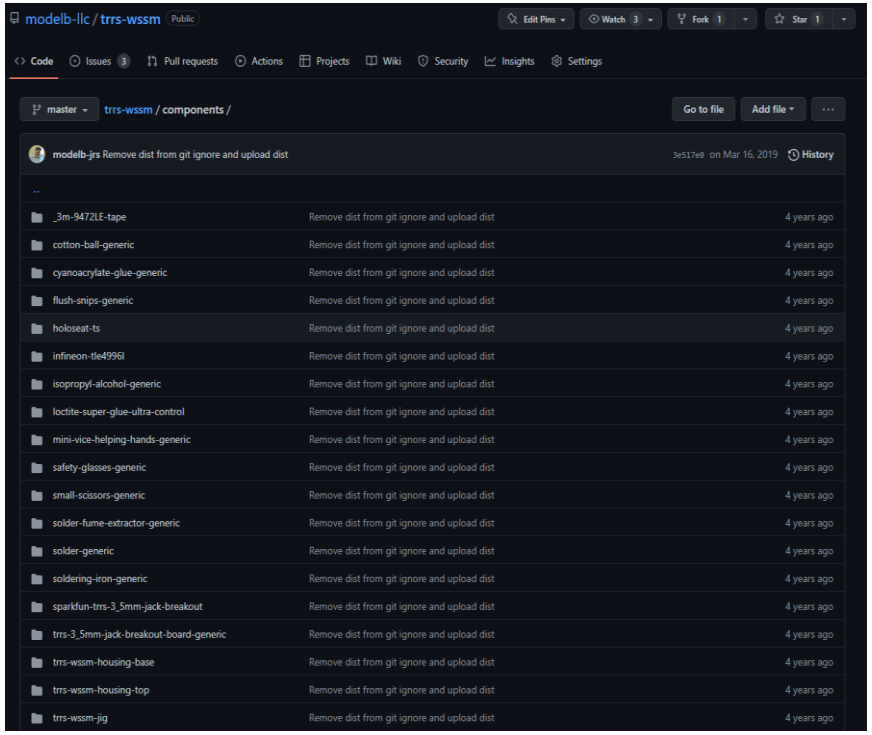


Table of Contents

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 – Version V0.1.6, 8/6/2021

Distributed OSHW Framework (DOF)

A. Simmons <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

thods have lasted through cvs,

https://github.com/Mach30/dof

What is the source of an OSHW project?

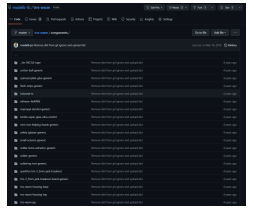
1. Bill of Materials (BoM) **Data** ← **Keep the Project Source DRY**
2. Assembly Instruction **Data** ← **Humans are the Compiler**
3. Supporting Materials ← **Except when the 3D Printer is the Compiler**

Branches, Forks, and Merges – Oh, My!

It’s Files All the Way Down

```
encoding_fallback_warning_issued = False
encoding_escape_warning_issued = False
def metadata_stream_to_writable_bytes(s):
    encodingStrategy = gitConfig('git-p4.metadataDecodingStrategy') or defaultMetadataDecodingStrategy
    fallbackEncoding = gitConfig('git-p4.metadataFallbackEncoding') or defaultFallbackMetadataEncoding
    if not isinstance(s, bytes):
        return s.encode('utf_8')
    if encodingStrategy == 'passthrough':
        return s
    try:
        s.decode('utf_8')
        return s
    except UnicodeDecodeError:
        if encodingStrategy == 'fallback' and fallbackEncoding:
            global encoding_fallback_warning_issued
            global encoding_escape_warning_issued
            try:
                if not encoding_fallback_warning_issued:
                    print("\nCould not decode value as utf-8; using configured fallback
                    print("\n(this warning is only displayed once during an import)")
                    encoding_fallback_warning_issued = True
                return s.decode(fallbackEncoding).encode('utf_8')
            except Exception as exc:
                if not encoding_escape_warning_issued:
                    print("\nCould not decode value with configured fallback encoding %s
                    print("\n(this warning is only displayed once during an import)")
                    encoding_escape_warning_issued = True
                escaped_bytes = b''
                # bytes and strings work very differently in python2 vs python3...
                if str is bytes:
                    for byte in s:
                        byte_number = struct.unpack('>B', byte)[0]
                        if byte_number > 127:
                            escaped_bytes += b'%'
                            escaped_bytes += hex(byte_number)[2:].upper()
                        else:
                            escaped_bytes += byte
                else:
                    for byte_number in s:
                        if byte_number > 127:
                            escaped_bytes += b'%'
                            escaped_bytes += hex(byte_number).upper().encode()[2:]
                        else:
                            escaped_bytes += bytes([byte_number])
                return escaped_bytes
```

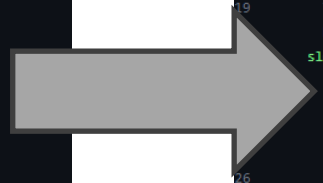
```
869  903          if skip_info:
870  904              if 'code' in entry and entry['code'] == 'info':
871  905                  continue
872      -        if 'desc' in entry:
873      -            entry['desc'] =
874      -        if 'FullName' in ent
875      -            entry['FullName'
     906  +        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  |  [*] => $e*
#
➤ Changes not staged for commit
#
#       deleted:  [1] _shared.sh
#      modified:  [2] assets/git_breeze/config.example.sh
#      modified:  [3] assets/git_breeze/config.sh
#      modified:  [4] assets/git_breeze/git_breeze.sh
#      modified:  [5] assets/git_breeze/lib/aliases_and_bindings.sh
#       deleted:  [6] assets/git_breeze/lib/git_file_shortcuts.sh
#      modified:  [7] assets/git_breeze/lib/git_repo_management.sh
#       deleted:  [8] assets/git_breeze/test/git_file_shortcuts_test.sh
#      modified:  [9] assets/git_breeze/test/git_repo_management_test.sh
#      modified: [10] assets/git_breeze/test/support/test_helper
#
➤ Untracked files
#
#     untracked: [11] assets/git_breeze/install.sh
#     untracked: [12] assets/git_breeze/lib/fallback/
#     untracked: [13] assets/git_breeze/lib/git_status_shortcuts.rb
#     untracked: [14] assets/git_breeze/lib/git_status_shortcuts.sh
#     untracked: [15] assets/git_breeze/test/git_status_shortcuts_test.sh
#
ndbroadbent@ndb-desktop ~/src/ubuntu_config [master±|1.9.2p290]$
```

Defining DOF – Modeling OSHW

```yaml
2-DataStructures/2-DataStructure.yaml: &ref_5
  name: Data Structure
  purpose: Define storage format for data used in an architecture
  template: |
    name: {{name}}
    purpose: {{purpose}}
    template: |
      {{template}}
    elements:
      - {{element}}
    derivedFrom: {{derivedFrom}}
  elements:
    - key: name
      type: string
      itemType: ''
      description: Human readable name of the Data Structure
      derivedFrom: []
    - key: purpose
      type: string
      itemType: ''
      description: Provides specification for an individual Data Structure
      derivedFrom: []
    - key: template
      type: string
      itemType: ''
      description: >-
        Liquid template of YAML representation of the data structure, suitable
        for use in manual or automated generation of the data structure
      derivedFrom: []
    - key: elements
      type: list
      itemType: 2-DataStructures/1-Element.yaml
      description: Definition of the data in the Data Structure
      derivedFrom: []
    - key: derivedFrom
      type: list
      itemType: string
      description: >-
        List of paths to model files that this Data Structure is related to by
        analysis results and design decisions
      derivedFrom: []
  derivedFrom: []
2-DataStructures/3-Reference.yaml: &ref_6
  name: Reference
  purpose: Provide data required to cite sources
```

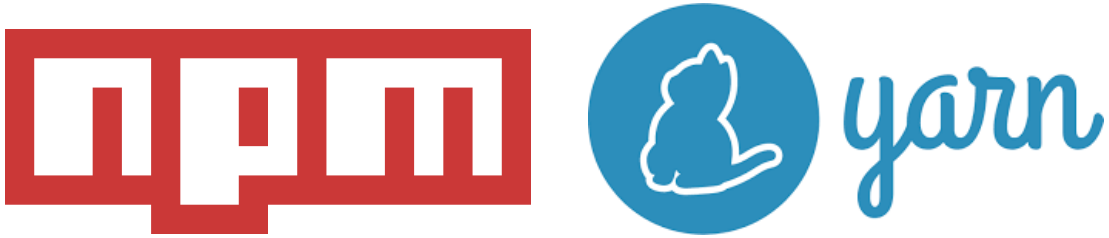


```yaml
id: https://w3id.org/Mach30/m30ml/blob/main/src/linkml/root.yaml
name: m30ml_root
title: Mach 30 Modeling Language - Root schema
description: |-
  Schema definitions for the Mach 30 Modeling Language (m30ml) Root Sy
license: https://creativecommons.org/licenses/by/4.0/
imports:
  - linkml:types
prefixes:
  linkml: https://w3id.org/linkml/
  schema: http://schema.org/
  m30ml_root: https://w3id.org/Mach30/m30ml/blob/main/src/linkml/root.yaml
default_prefix: m30ml_root
default_range: string

classes:
  Element:
    description: :-
      The basic construct of an m30ml model.  All other classes inherit from Element such that a model is a collection of Elements.
      Elements may own other Elements via the *body* slot.
    slots:
      - id
      - shortName
      - name
      - body

slots:
  id:
    identifier: true
    required: true
    description: :-
      An element's [Globally Unique Identifier (GUID)](http://guid.one/guid) in a model.
      Tools may choose the GUID version to use (language default is v4).
    close_mappings:
      - schema:identifier
    # from https://www.geeksforgeeks.org/how-to-validate-guid-globally-unique-identifier-using-regular-expression/
    pattern: "^[{]?[0-9a-fA-F]{8}-([0-9a-fA-F]{4}-){3}[0-9a-fA-F]{12}[}]?$"
  shortName:
    required: true
    description: :-
      An abbreviated name useful for referring to the element.
      Conforms to the NPM package.json *name* [format](https://docs.npmjs.com/cli/v8/configuring-npm/package-json#name).
    close_mappings:
      - 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