D3 - A system for recording complex experiments with an extension of SigMF

Ankur Mohan AMOHAN@IQT.ORG
2107 Wilson Blvd. Suite 1100 Arlington, VA 22201 USA

Ravi Pappu RPPAPU@IQT.ORG
2107 Wilson Blvd. Suite 1100 Arlington, VA 22201 USA

Sean Shadmand SSHADMAND@IQT.ORG
800 El Camino Real Suite 300 Menlo Park, CA 94025 USA

Abstract

The Signal Metadata Format (SigMF) is an emerging data interchange format (Hilburn, 2018) that specifies a way to describe recorded digital signal samples with metadata written in JSON (ECMA-404). We broaden this idea to describe *experiments*, which are collections of related signals. and the associated sensors that collect them.

We were motivated to develop this extension by a real-world problem. DARPA's Aerial Dragnet Program (DARPA, 2016), observes Unmanned Aerial Vehicles (i.e., drones), with multiple simultaneous sensors with the goal of several answering analytic questions (e.g., what is the identity of the drone?). Given the combinatorial explosion of possible experimental observations and the large data sizes involved, we needed a simple way to catalog and discover relevant data sets without having to download them first.

This paper describes our data collection and distribution use case, the SigMF-inspired extension we developed, and our experience with using the resulting JSON to build an experimental data browser christened D3 - for Drone Data Distribution system.

1. Scenario and requirements

In this section we introduce the data collection scenario and the requirements that led us down the path of extending the idea behind SigMF.

Proceedings of the 1st GNU Radio Conference, Copyright 2016 by the author(s).

1.1. Data collection scenario

During Aerial Dragnet data collection, a number of consumer drones were flown multiple times in a variety of urban environments. Each flight, from takeoff to touchdown, represents one experiment. Note that an experiment can consist of multiple drones fliying at the same time, with multiple sensors observing them. Data from each experiment was collected by an array of sensors active during the experiment. This resulted in a huge volume of heterogenous data (and corresponding metadata) generated per experiment which needed to be easily located, understood and downoaded.

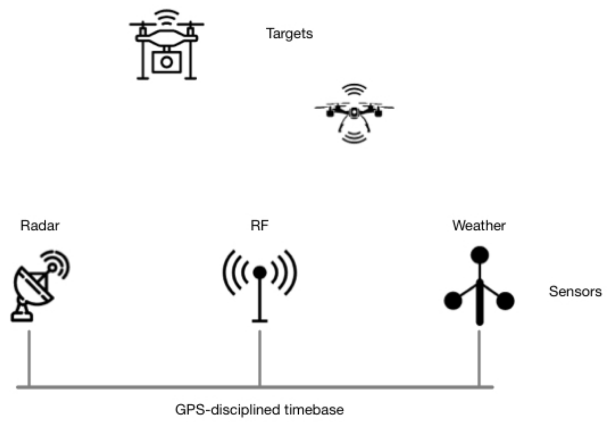


Figure 1. D3 data collection scenario. There can be any number of drones or targets visible, and there can be any number of sensors simultaneously collecting data, all of whom are synchronized to the same GPS-disciplined clock.

In addition to the large volume of data, another complicating issue is the lack of a clear correspondence between an experiment and data files. Certain sensors are naturally only active during an experiment. For example, the GPS and Inertial Measurement Units on board a UAV are activated when the UAV is turned on and are automatically deactivated when the UAV is turned off. Thus the corresponding data files are automatically associated with the experiment and a many to one mapping between the data files and the experiment exists. Other sensors are active during the entire data collection process. The output of such sensors is one file that spans multiple experiments or multiple files that lack a clear correspondence with the experiments. See Figure 2

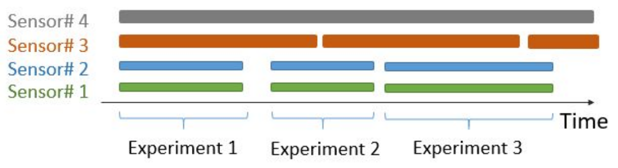


Figure 2. Sensor 1, 2 are automatically activated/deactivated when an experiment starts/ends. Thus their output data naturally corresponds to experiment time boundaries. Examples of such sensors are onboard UAV IMUs and GPS units. Sensor 3 writes its data to the disk at regular time intervals not synched with experiment boundaries. For example, a video recorder that writes the video file to the disk automatically once the file exceeds a certain size. Sensor 4 collects data continuously during the course of the data collection and saves its data at the end of the data collection into one large file. For example, an RF/audio background sensor

1.2. State of prior art

Sensors vary widely in the type of data they collect, how the data is processed by the sensor and the formats used to output the sensor data. Vendor provided drivers and other software tools are typically required to decode the sensor data and interpret it. Sensor operating parameters and ground truth information is also required to properly calibrate the data.

No industry standard for systematically organizing such heterogenous sensor data exist exist currently. An experiment designer comes up with their own scheme to organize and describe the data. These schemes typically consist of custom filenames, custom directories for the data files, groundtruth files, software tools necessary to decode the sensor data, sensor drivers and so on. Information about how the data is organized is usually scattered across multiple documents. A user must spend considerable effort to understand the data organization and locate and decipher the slice of data in which they may be interested. Furthermore, each user is likely to have their own preferred method to organize their slice of data. Due to lack of consistency in organizing and describing the data, it can't be easily ingested by data inspection and visualization tools.

Prior to the adoption of our scheme, Aerial Dragnet sensor data and related metadata was organized in a custom manner. There was no consolidated document that contained all the information necessary to understand the data. There were various folders that contained software tools, sensor operating parameters, ground truth data and a set of documents that described how all this information should be used. A user had to read multiple documents and keep track of the location of these documents to understand the data.

A second problem was related to how sensor data files were named. The experiment information was encoded in the filename - for example ”TestSite_2017Apr10_Run1_UAS1_GroudTruth_GPS1.dat”. This may appear to be a good idea at first as the filename offers information about the characteristics of the experiment that generated it at a glance. However this scheme has many drawbacks. First, for a sufficiently complex experiment, comprehensive experiment information can't be embedded in the filename if the filenames are to be kept a resonable length. Therefore, there must be an acconpanying metadata document that describes the details left out in the filename. Second, the filename itself can't be modified without running the risk of orphaning the file, i.e., removing the link between the file and its source. Even if the name modification doesn't result in orphaning the file, it may remove or alter critical information about the file that could confuse the user. For instance, a simple inadvertent modification of ”Run1” to ”Run2” in the filename above would lead the user to think the data came from the wrong experiment.

1.3. System requirements

At the time of our involvement in this project, the data had already been collected. Our task was to devise a system to efficiently organize and distribute this data so it could be made available to performers in the DARPA program relatively quickly. Facing this time constraint, we could not conduct a comprehensive survey of other data organization schemes. We decided to adapt a SigMF inspired scheme given our prior familiarity with SigMF and its growing adoption in describing large signal datasets. Our system requirements are listed below.

1. Single point of contact: We would like to have a single document that completely describes the experiment *i.e.,* what was done and where the data is stored.

2. Separation of data storage from metadata: The system shouldn't impose any storage requirements on the data and metadata. It should be possible to store the data files whereever appropriate with links to the files contained in the metadata

3. Access control by type of user: The system should allow for different levels of access control to the data. One user can have access to certain data files, but not

to others. The access control information is maintained separately by system administration and in not included in the metadata

4. Adaptive user interface for dynamic metadata: The user interface for the data should be data driven and update automatically as new fields are added (for example, new sensors and targets)

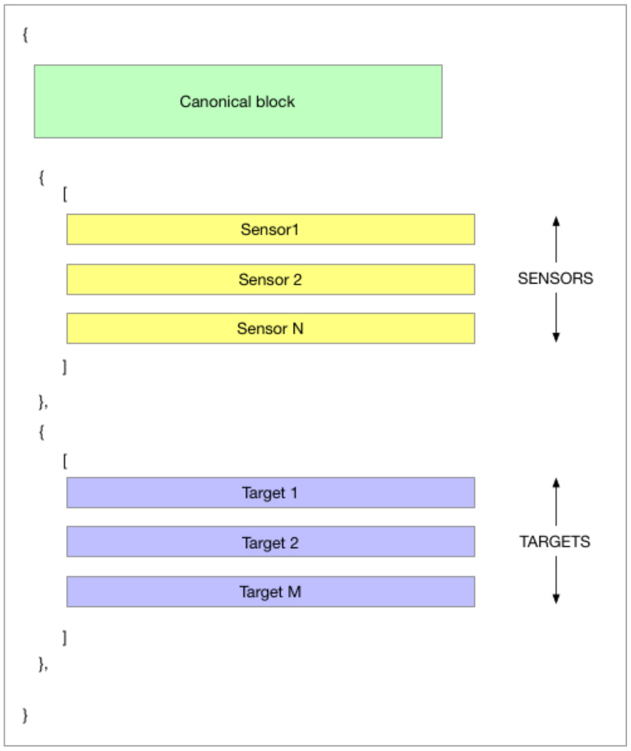


Figure 3. Anatomy of a D3 JSON file. See figure 4 for details

2. Extending SigMF

2.1. Anatomy of an experiment

During Aerial Dragnet data collection, a number of consumer drones were flown multiple times in a variety of urban environments. Each flight, from takeoff to touchdown, represents one experiment. Note that an experiment can consist of multiple drones flying at the same time, with multiple sensors observing them. Data from each experiment was collected by an array of sensors active during the experiment.

To describe such experiments, D3 features a single JSON file corresponding to an experiment (also known as a ”run”) that includes all the information necesary to understand the data collected during the run. A *run.json* file contains key-value pairs in JSON format that provide general information about the run, the sensors that were active during the run and the targets used during the run (about which information was collected). Thus, each *run.json* file has three sections, as described below and illustrated in Figure 3 and Figure 4.

2.2. Run-Specific Canonical Metadata

A *run.json* begins with a unique *Run ID*, followed by Run-specific metadata such as location, time, environmental conditions, participants etc., that apply to the Run as a whole. The units used for the metadata are made part of the meta data field. Any additional information about the unit format is described in a ”comment” field.

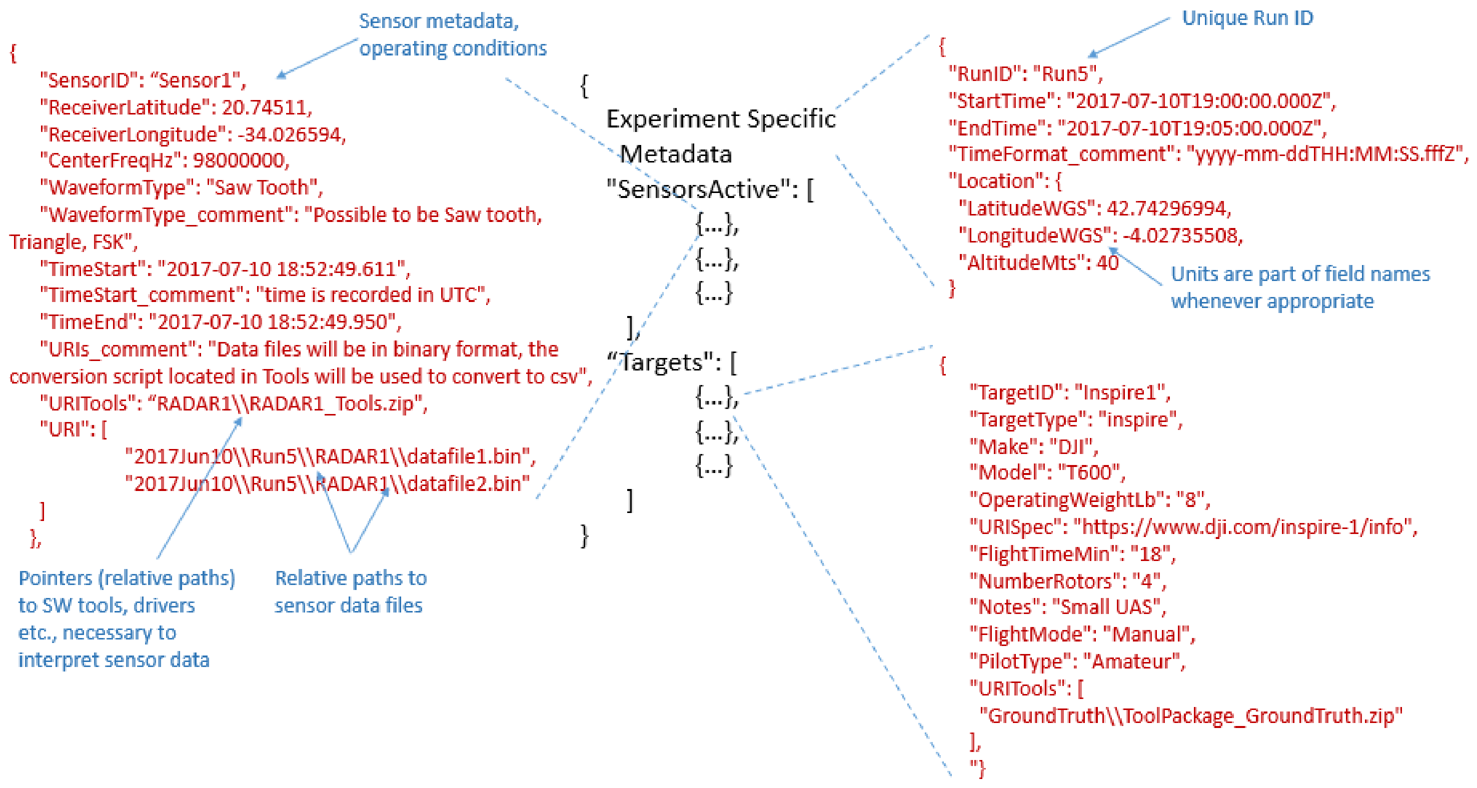
2.3. Sensors Array

General information about the Run is followed by a ”Sensors” array. Elements of this array contain information about sensors that were collecting data about one of more targets during the Run. This information includes operating parameters of the sensor - for example, the Center Frequency for a RADAR sensor, the location of any ground truth, software driver and documentation files necessary to decipher the sensor data; and the relative links to the data files collected by the sensor.

Since all necessary sensor metadata is included in the Sensors array entries, there are no requirements or special naming conventions for the data filenames. The filenames can be modified and the files be moved to different folders as long as the relative links in the parent JSON document are kept consistent. There is no uniqueness requirement either - files in different folders can have the same names. It's easy to check for data consistency by verifying that the pathnames in the *run.json* files point to actual files on the disk. The user can be warned about any inconsistencies.

The units for each piece of information are embedded in the field name and additional information about the data format is provided in the corresponding ”comment” field. This helps prevent bit rot where important information about pieces of data can be lost over time. Preventing bit rot is a design goal of SigMF. The information about each sensor in the Sensors array is comprehensive so that the user has all the information needed to understand the data in one place. This follows the ”Single Point of Contact” design principle we established earlier. In our drone data collection use case, each sensor was collecting information about all the drones flying during the data collection. In other cases, a given sensor could only be collecting information about a single target or a subset of targets - in this case, a ”targetID” field could be included in the elements of the Sensors array indicating the target about which that sensor was collecting data.

Certain sensors are naturally only active during an exper-



```
{
Experiment Specific
 Metadata
"SensorsActive": [
        {…},
        {…},
        {…}
  ],
"Targets": [
        {…},
        {…},
        {…}
  ]
}
```

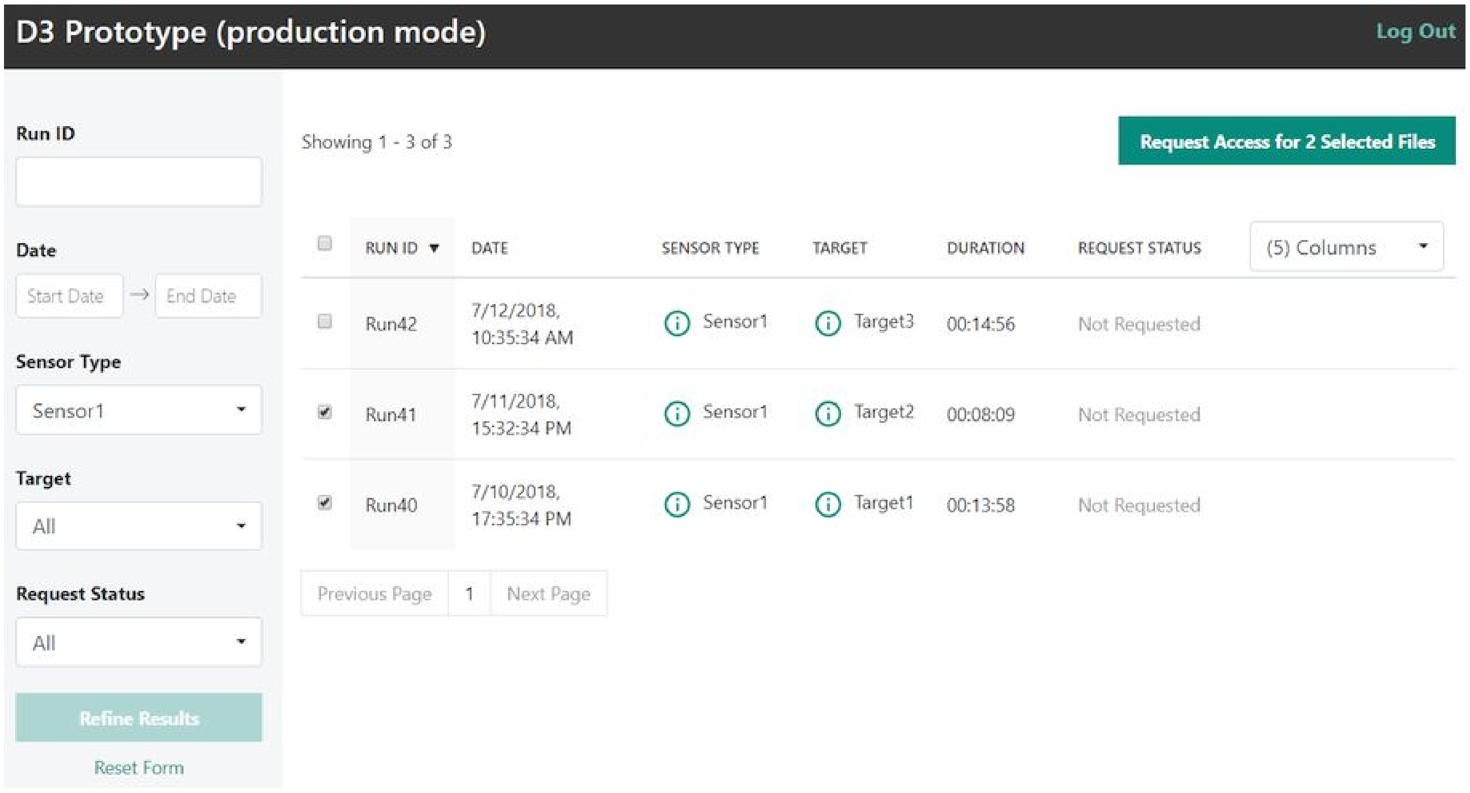
```
{
    "SensorID": “Sensor1",
    "ReceiverLatitude": 20.74511,
    "ReceiverLongitude": -34.026594,
    "CenterFreqHz": 98000000,
    "WaveformType": "Saw Tooth",
    "WaveformType_comment": "Possible to be Saw tooth,
Triangle, FSK",
    "TimeStart": "2017-07-10 18:52:49.611",
    "TimeStart_comment": "time is recorded in UTC",
    "TimeEnd": "2017-07-10 18:52:49.950",
    "URIs_comment": "Data files will be in binary format, the
conversion script located in Tools will be used to convert to csv",
    "URITools": “RADAR1\\RADAR1_Tools.zip",
    "URI": [
             "2017Jun10\\Run5\\RADAR1\\datafile1.bin",
             "2017Jun10\\Run5\\RADAR1\\datafile2.bin"
    ]
  },
```

```
{
  "RunID": "Run5",
  "StartTime": "2017-07-10T19:00:00.000Z",
  "EndTime": "2017-07-10T19:05:00.000Z",
  "TimeFormat_comment": "yyyy-mm-ddTHH:MM:SS.fffZ",
  "Location": {
    "LatitudeWGS": 42.74296994,
    "LongitudeWGS": -4.02735508,
    "AltitudeMts": 40
}
```

```
{
    "TargetID": "Inspire1",
    "TargetType": "inspire",
    "Make": "DJI",
    "Model": "T600",
    "OperatingWeightLb": "8",
    "URISpec": "https://www.dji.com/inspire-1/info",
    "FlightTimeMin": "18",
    "NumberRotors": "4",
    "Notes": "Small UAS",
    "FlightMode": "Manual",
    "PilotType": "Amateur",
    "URITools": [
      "GroundTruth\\ToolPackage_GroundTruth.zip"
    ],
    "}
```

Sensor metadata, operating conditions
Unique Run ID
Units are part of field names whenever appropriate
Pointers (relative paths) to SW tools, drivers etc., necessary to interpret sensor data
Relative paths to sensor data files

Figure 4. Anatomy of a D3 JSON file in detail. See text for details about each section



Showing 1 - 3 of 3

☐	RUN ID ▼	DATE	SENSOR TYPE	TARGET	DURATION	REQUEST STATUS
☐	Run42	7/12/2018, 10:35:34 AM	Sensor1	Target3	00:14:56	Not Requested
✓	Run41	7/11/2018, 15:32:34 PM	Sensor1	Target2	00:08:09	Not Requested
✓	Run40	7/10/2018, 17:35:34 PM	Sensor1	Target1	00:13:58	Not Requested

Figure 5. D3 web client

iment. For example, the GPS and Inertial Measurement Units on board a UAV are activated when the UAV is turned on and are automatically deactivated when the UAV is turned off. Thus, the corresponding data files are automatically associated with the experiment. Other sensors are active during the entire data collection process. The output of such sensors is one file that spans multiple experiments or multiple files that lack a 1-1 correspondence with the experiments. Such cases can be handled by embedding the beginning and ending data pointers along with the data file names in the *run.json* files. See figure 2 as an example.

2.4. Targets Array

The Sensors array is followed by the targets array. The elements of this array provide information about the targets about which information was collected during the experiment. The structure of the Targets array is similar to the Sensors array. Each element of the array provides general information about the target, for example the make and model of a drone followed by any data files, such as the log file for a drone flight.

3. Putting it all together

The complete D3 system consists of three components. First, a web client that enables users to inspect the sensor data, perform queries such as ”show me all data collected by a certain sensor where a certain drone was flown” and request access to data slices of interest. See figure 5 for a screenshot of the client. Second, an admin console where an administrator can add/remove D3 users and approve/deny access requests. Lastly, a download manager application helps the user securely download the data to which they have access keeping track of data the user may already have downloaded previously. This is an important consideration when downloading huge amounts of data. The information presented by the client is automatically pulled from the *run.json* files. Sensor and target types don’t need to be hard coded into the system, they are automatically populated from the JSON files as new sensors and targets are added.

Visualizers for different pieces of information collected by the sensors can be easily added. For instance, the system could pull the flight path coordinates of a drone flight from the groundtruth file and display the flight path for a run in a tooltip window next to a target name. Similarly, helpful information about the sensors can be pulled from the sensor metadata and displayed next to the sensor name.

The system is currently deployed on Amazon Web Services Govcloud.

4. Conclusion and Future Work

We have described an extension of SigMF to include the notion of Experiments, which are collections of signals and sensors that collect them. We were motivated to do this out of a problem arising in our work. Our intent is to find a way to merge this idea into the SigMF repository in the near future.

References

DARPA. Aerial dragnet. Technical report, 2016.

ECMA-404. The JSON Data Interchange Syntax. Standard, ECMA International, Geneva, CH, December 2017.

Hilburn, B. The signal metadata format (sigmf). Technical report, 2018.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.