Received 31 August 2023, accepted 29 September 2023, date of publication 4 October 2023, date of current version 11 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3322105

APPLIED RESEARCH

A Real-World Dataset Generator for Specific Emitter Identification

BRAEDEN P. MULLER[1,2], (Member, IEEE), LAUREN J. WONG[2,3], WILLIAM H. CLARK IV[1], (Member, IEEE), AND ALAN J. MICHAELS[1,2], (Senior Member, IEEE)
[1]National Security Institute, Virginia Tech, Blacksburg, VA 24060, USA
[2]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24060, USA
[3]AI Laboratory, Intel Corporation, Santa Clara, CA 95054, USA

Corresponding author: Braeden P. Muller (braedenm@vt.edu)

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

ABSTRACT Generating high-quality, real-world, well-labeled datasets for radio frequency machine learning (RFML) applications often proves prohibitively cumbersome and expensive, leading to the low availability of high-fidelity, low-cost datasets. Specific emitter identification (SEI) in particular requires a hardware setup capable of supporting transmitting using many different radios, while automated modulation classification (AMC) performance is primarily driven by SNR, channel effects, and the similarity of modulation types. These factors give rise to the need for scalable methods of inexpensive dataset generation. This paper describes the design considerations and a proof-of-concept implementation of a blind user reconfigurable platform capable of creating SEI and AMC datasets throughout a variety of real-world conditions. This paper additionally describes the reliability and performance of the platform relative to existing real-world data generation methods and compares generated datasets to those already present in the literature. This work also describes the software post-processing steps taken to isolate, label, and cull captured data and transform these into a high-quality dataset.

INDEX TERMS Specific emitter identification (SEI), automated modulation classification (AMC), RF fingerprinting, radio frequency machine learning (RFML), radio emitters, real-world dataset generation, dataset generation.

I. INTRODUCTION

Machine learning (ML) has resulted in advances in the state-of-the-art in the fields of computer vision, voice recognition, natural language processing, medical imagery, and finance. Notably absent from this list is the area of radio signal processing [1]. Applying ML to radio applications, referred to as radio frequency machine learning (RFML), shows increasing promise for the augmentation of capabilities for signal detection, classification, and estimation [2], [3], enabling enhanced security and radio spectrum sharing techniques [4], [5]. ML is used to build correlations and draw relationships between large sets of data. However, this data is only useful when it is relevant and tailored to the resulting application. Depending on the application, application-specific data can often be difficult and costly to procure [6].

Internet of Things (IoT) applications particularly benefit from additional security measures made possible through RFML [7], [8]. The continuous expansion of IoT networks presents a growing attack surface even while adhering to best security practices [7], [9]. As the number of lightweight and rudimentary authenticated wireless devices within a network grows, the susceptibility to, and likelihood of, common network attacks also grows [7]. Increasing size of a network further increases the attack surface for node impersonation, eavesdropping, or denial of service (DoS) attacks that compromise the confidentiality, integrity, and availability of the network.

Solutions to the problem of specific emitter identification (SEI) present a possible mitigation to these concerns. SEI is a process of determining the identities of individual emitters by

The associate editor coordinating the review of this manuscript and approving it for publication was Shuangqing Wei.

comparing radio frequency (RF) fingerprints. These RF fingerprints, which are the unavoidable distinguishing features that arise from the analog components in each emitter, are orthogonal to the actual information being transmitted [10], [11], [12], [13]. The main goal is to augment the security provided by software and firmware measures, which typically occur at Medium Access Control (MAC) or Network layers and are vulnerable to manipulation and impersonation, by incorporating observation of nondeterministic and unclonable aspects of the hardware. Through the measurement of subtle and unintentional modulations on pulse (UMOP) caused by manufacturing tolerances and non-idealities, SEI systems aim to identify features that indicate the identity of each wireless transmitter [14], [15], [16] that may be used as an additional layer of identity confirmation in IoT networks on top of traditional cryptographic authentication protocols.

SEI and RF fingerprinting have applications in a broader security context, such as wirelessly detecting anomalies which could indicate an adversary attempting to mimic or impersonate a node using captured legitimate security credentials or detecting attempts of intrusion and spoofing with forged MAC addresses [17], [18]. Additionally, SEI and RF fingerprinting can be used in military applications, where they are useful in early warning systems and locating and tracking emitters [19]. These models also have utility for cognitive radio, where a software defined radio (SDR) may adjust its behavior according to its RF context [20]. Context-dependent behavior is useful for enforcing dynamic spectrum access (DSA) rules for future wireless communication protocols that are designed to more efficiently utilize allocated RF spectrum resources [15].

One of the earliest successful approaches to RF fingerprinting, PARADIS, was intended to be used as an extra layer of security for 802.11 networks [16]. A common feature of this and similar approaches to both RF fingerprinting and SEI is their tendency to use hand-crafted algorithms that rely on expert features to identify transmitters. These expert features include estimated frequency, received power, phase error, I/Q offset, SYNC correlation, packet inter-arrival time [16], [17], [20], [21], [22], and derived features of the power-on transient energy envelope [23]. These expert features that are often used, however, suffer from a lack of flexibility that impedes future progress in development of these algorithms. Instead, deep learning (DL) and neural network (NN)-based approaches that operate on raw I/Q data may be preferred since they can offer greater flexibility in identifying features of a signal that a designer would otherwise be unable to notice or manually describe [7], [9], [17], [19]. Part of this desire for ML-based features derives from the difficulty to precisely model the non-ideal analog circuitry in the RF signal chain over all ranges of environmental and operating conditions. Additionally, attempts to spoof or subvert an SEI or RF fingerprinting unit will likely account for first order RF nonlinearities, making the higher-order effects of interest for detection of anomalies or mimicked signals.

DL and NN approaches, including convolutional NNs (CNNs) and recurrent NNs (RNNs), require a great deal of training data to function adequately. Unlike in the areas of audio or image recognition, standardized datasets for RFML application spaces are largely lacking, partially due to the extreme variances in potential transmitter hardware, channel conditions, and receiver assumptions. A majority of previous works address this problem by creating their own dataset [14], [15], [16], [18], [20], [22], either simulated, real-world captures, or augmented [8]. Simulated data can be created quickly and efficiently, but engineers lack the ability to completely model all non-idealities present in any given transmission. Real-world data must be similar in nature to the deployment environment, which is often costly, time consuming, and potentially impossible to gather [24]. An observed risk in prior SEI work with ML techniques is the learning of the differences in propagation channel between emitters rather than the RF hardware characteristics, leading to the need to generalize that propagation path in future experimentation [25], [26].

To cope with limited availability, existing real-world datasets are often augmented to increase the amount of useful data available and to generalize across channel conditions and propagation environments [27], [28]. For SEI applications, it has been observed that differing channel conditions and propagation environments between training and evaluation datasets can lead to a significant degradation of model performance [29], yet many of the augmentations do not apply since the RF fingerprints cannot be synthetically modified without distorting the learned behaviors. For AMC applications, augmentation methods can be employed to create much larger effective datasets. As desired, various augmentation methods to produce data suitable for the training of channel-insensitive models remains an active area of research [30], [31].

Efforts to collect real-world data are confounded by practical scaling difficulties; while it is easy and inexpensive to collect a small amount of data from two radios in one channel environment, it is exceedingly difficult and expensive to collect a large quantity of data from several hundred radios in a wide variety of channel environments. In training generalized SEI models, captures are required from possibly hundreds or thousands of emitters so that it may accurately discriminate between known and unknown emitters; this scaling problem is a distinct challenge. Furthermore, in a high-quality dataset, the emitter source of every transmission must be labeled with perfect accuracy, lest mislabeled entries poison the overall quality of the dataset [32]. The problem of coordinating labeling becomes more complex as the number of possible permutations of receivers, transmitters, and channel environments increases. In contrast to existing ML applications, such as those in image processing, a robust implementation of an SEI model requires capture of non-idealities and subtle differences in received signals that could otherwise be discarded as noise. The closest existing modality of comparison would be natural language processing (audio) with the intent to identify the speaker, but not necessarily the speech.

Due to these considerable impediments, it is common for RFML researchers to use MATLAB, GNURadio, or other similar toolkits to generate simulated datasets [1]. DeepSig Inc. hosts a few popular freely-available simulated datasets for AMC [33], but they are from early academic research, are no longer maintained, and even the authors suggest simulating new data or using real-world captures. Northeastern University's Institute for Wireless Internet of Things hosts 5 categories of datasets for RF fingerprinting with captured, simulated, and augmented data. Included in these are data from Wi-Fi, LTE, and 5G stationary emitters and mobile emitters that are attached to airborne UAVs [25], [29], [34], [35], [36].

Perhaps the highest quality existing real-world dataset is the DARPA RFMLS dataset [37], which has over 103 million labeled transmissions from over 53 thousand Wi-Fi devices and over 3.5 million transmissions from over 10 thousand ADS-B transponders. To generate a dataset with the same number of examples, it is estimated that it would take a single instance of this approach with one receiver around 82,000 hours. This estimate is calculated based on the performance characteristics of the proof-of-concept model and discussion for how this was derived is located in Section IV-B1. The scalability of the platform means that this duration is easily reduced by running several multi-receiver setups in parallel. The DARPA RFMLS dataset is subject to U.S. export restrictions and is therefore exclusively available to a narrow pool of researchers.

More widely available are the datasets cataloged by the CRAWDAD resource [38], a majority of which are either not properly labeled for SEI, not large enough to capture diverse channel conditions, or have too few labeled emitters to be applicable to a general SEI model. Available on this resource is the *uw/sigcomm* dataset [39], that is a collection of miscellaneous 802.11 wireless monitoring traces along with corresponding *tcpdump* data. A comparatively larger dataset for Wi-Fi is "Massive-Scale I/Q Datasets for Radio Fingerprinting" [40], which has 2 TB of collected data from 20 sample devices.

For IoT and low-cost devices, the DroneRF dataset [41] is a typical example of a small dataset that has 227 different collection segments from 3 different commercially-available drones. The Device Identification dataset, featured by the IEEE Communications Society Machine Learning For Communications Emerging Technologies Initiative [42], is intended to serve as a common benchmark for IoT SEI models, but has seen limited adoption thus far with only one mention in a survey paper [43], but has not yet been cited by any other works. The IoT SENTINEL dataset [44] features captures from 31 smart home IoT devices. The IoT device identification dataset [45] provides 50 GB of 863-870 MHz radio spectrum measurements from different rooms of the same building. A common observation between the freely-available datasets is their tendency to include data on the order of 10s of GB from on the order of 10s of devices.

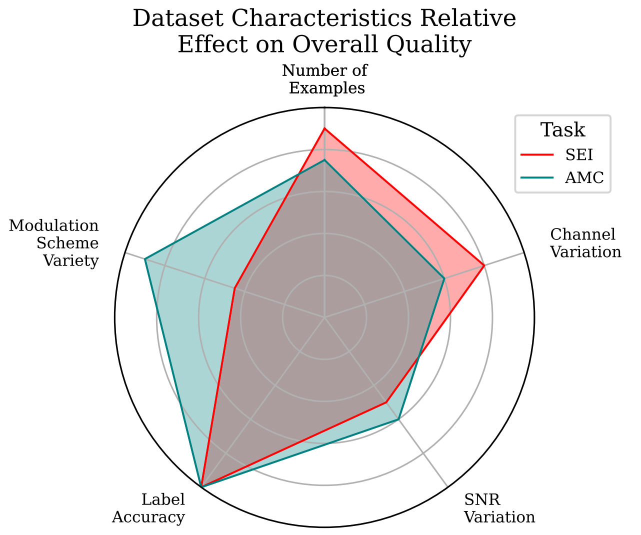

FIGURE 1. General visual overview of the parameters contributing to the quality of a dataset for the two RFML tasks of this work's focus. These are not direct relationships, but rather serve as a general intuition how "valuable" each aspect may be for the creation of a performant model.

From this survey of datasets and past work assessing the quality of datasets [6], Fig. 1 shows a quick-guide for a general impression of the main takeaways on how various characteristics of a dataset affect its "goodness" for SEI and AMC applications. It should be stressed that a good dataset is well-rounded when it comes to these criteria and the figure instead means to communicate that any sacrifices when it comes to absolute label accuracy is particularly degrading to the overall quality of the dataset and the resulting performance of models trained on it.

The Blind User-Reconfigurable Platform (BURP), the subject of this work, is not a dataset, but rather the design of a semi-portable hardware testbed that provides the means of creating robust datasets for around 100 software-defined devices over a long period of time and in a variety of environments. BURP is intended to automate parts of the process of transmitting, receiving, and cataloging RF data. This processing is performed in a single-blind fashion, where a report of groundtruth for the emitters, including framing sequences and metadata characteristics of the signals, is retained for scoring the performance of SEI algorithms. The platform hosts low-end, inexpensive transmitters such as those used for IoT applications, but could easily be extended to host other radios of interest, potentially even wrapping the radios in a shared application programming interface (API) that enables heterogeneous architecture or co-channel signal environments.

The choice of low-cost YARD Stick One devices, discussed in Section II-A, comparable to those used in IoT applications is especially useful for the later development

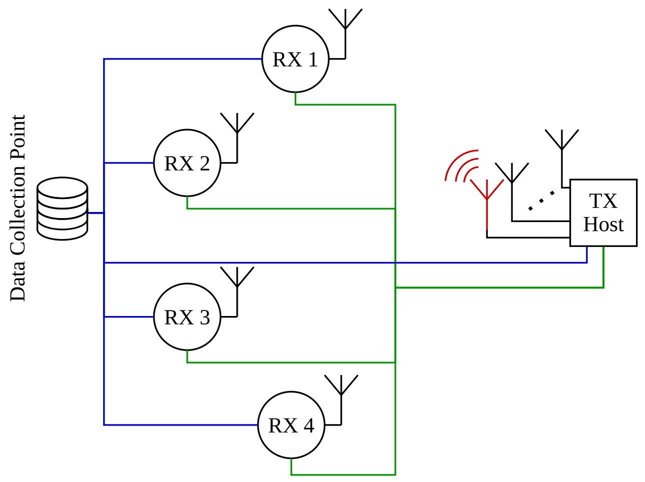

FIGURE 2. High-level diagram of the major components of BURP. The transmitter host manages an array of transmitters, and is connected to the receiving "collection nodes" through the control (green) and data (blue) back-planes.

of SEI algorithms for the IoT space. These SEI algorithms could possibly be used as a physical-layer security measure to detect and prevent the introduction of counterfeit devices. Large datasets, such as those produced by BURP, are imperative to compensate for the vast quantities of devices that these models will encounter in diverse propagation environments. In the creation of BURP, the goal is to create well-labeled datasets with higher population sizes than what is otherwise available in existing literature. The same core software infrastructure presented in this work offers extension to support other RF emitter types, whether better quality RF transmitters, RADARs, or other devices.

BURP overcomes the challenge of coordinating a large number of devices and multiple receivers to transmit, collect, and label data. The system's chief design goals are hands-off operation and scalability. Therefore, to improve stability, the system features remote monitoring, automatic error checking, and recovery procedures which include the ability to control the power available to connected devices. A high-level diagram of the proof-of-concept BURP setup is shown in Fig. 2. The individual components of the platform are explained in detail in later sections. The use of the USB protocol and the incorporation of software abstraction layers enables the system to generalize for a variety of communications, RADAR, and other RF-based applications.

The paper is structured as follows. In Section II, the requirements and desirable characteristics of the platform are discussed, culminating in an overall experimental framework in Section III. Section III focuses on the design of the experimental hardware setup, addressing many of the lessons learned and constraints of low-cost components. Section IV describes the testing and validation results, performance measurements, and support for potential future RFML experimentation, including, but not limited to, modulation recognition and specific emitter identification. Section V gives a summary and conclusions, along with plans for future work that will include open publication of a series of RFML datasets using the BURP machine.

II. GENERATION REQUIREMENTS

The primary requirement of BURP is to provide a modular platform that is extensible to support an arbitrary number of low-cost wireless devices with an abstraction layer to allow the user to easily generate datasets consisting of short RF energy bursts to their specifications. This is to reduce the time, complexity, and monetary cost of generating a large amount of data for use in development, training, and testing of RFML systems. Moreover, many of the anticipated RF fingerprint features occur during the turn-on and turn-off transients of the RF burst, lending a preference towards the ability to generate short bursts. As a proof-of-concept, we chose a reconfigurable platform, supporting up to 120 low-cost USB-based software-defined radios (SDRs). This limitation is a consequence of the USB protocol's indexing, which only supports up to 127 devices on a bus, including hubs [46].

BURP is intended to be flexible and configurable enough to support the creation of datasets for a variety of RFML experiments. More specifically, BURP is capable of producing data suitable for a wide range of RFML use-cases, including signal detection, AMC, and SEI, with varying channel conditions, center frequencies (CFs), data rates, sampling rates, burst contents, burst lengths, transmit powers, and modulation schemes. Currently of interest are experiments related to evaluating RF transfer learning (TL) performance under such changes in channel conditions, transmitter/receiver hardware configuration, and use-case, extending recent research [47], [48] from synthetic to captured data. Such experiments aim to identify how the channel, transmitter/receiver hardware, and use-case impact learned behavior and facilitate or prevent successful TL. Future work facilitated by BURP data also includes development of frequency and data rate agnostic RFML algorithms, and examining the impacts of transients, preambles, temperature, humidity, and RF front ends on RFML performance.

A. TRANSMIT

For transmitting RF bursts, the radio chosen was the YARD Stick One (YS1), based on a Texas Instruments CC1111 MCU. This was chosen for its relatively low unit cost, and its ability to transmit on a range of CFs (300-348 MHz, 391-464 MHz, 782-928 MHz) with different modulation schemes (ASK/OOK, GFSK, 2-FSK, 4-FSK, MSK), and at a range of power levels (-30 to +10 dBm) [49], [50]. The central component which the YS1 is constructed around, the Texas Instruments CC1111Fx MCU, is responsible for communication with the transmitter host through its USB header, control, and signal modulation [51]. A diagram of the YS1, its layout, and its major components is shown in Fig. 3. The BURP transmitter host system should provide power and control to the entire array of YS1 transmitters used

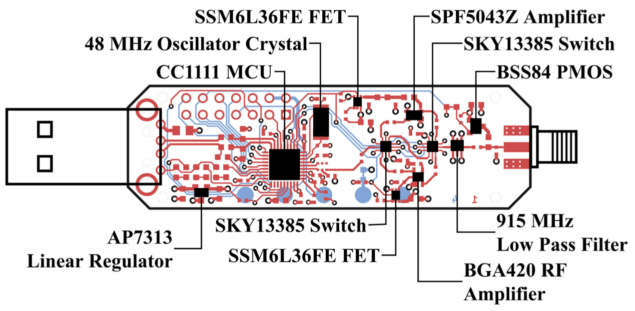

FIGURE 3. Layout and components of the YARD stick one wireless test tool.

for data generation. Using the YS1 is just a proof of concept, as the platform should be able to support more than just these specific transmitters. The platform, therefore, needs to be constructed in such a way that it may be adapted to support any such low-cost SDR device or wireless emitter with a USB connector.

When the user wishes to generate data according to their specifications, such as any arbitrary combination of supported center frequencies, bandwidths, and modulation schemes, the transmitter host should be able to read and interpret the given specifications and use this to build a "groundtruth" for a particular run. This groundtruth file includes relevant information such as framing sequences, timestamps, payloads, CFs, data rates, modulation schemes, etc. The metadata labels in the groundtruth file are necessary for training and evaluating the performance of SEI models.

An important consideration for the transmitter host is the detection, identification, and amelioration of errors and hardware failures, which were much more prevalent than originally expected in the case of the YS1. This should extend to include capabilities to notify the user of failed transmitters and either exclude failed transmitters from the program flow, or automatically attempt to recover failed devices and continue normal operation. The transmitter host, therefore, needs to be aware of its hardware status at all times, what transmitters are connected, and where they are connected.

B. RECEIVE

The receive end of the BURP platform needs to host the actual data collection facilities. For this purpose, there should be multiple receiving-end "collection nodes" (CNs). These CNs need to be capable of hosting much higher resolution radios than the transmitter host, such as USRPs or other SDRs, and be portable enough to easily be moved, re-oriented, and re-configured to create datasets more generalized over channel conditions and multi-path effects from different rooms or locations within a room.

Each CN handles its own data collection, labeling, tagging, and storage. This process should happen in coordination with the transmit end of the platform. Collections should begin and end when prompted by the transmit end of the system through a control back-plane, where the transmit end will be expected to provide information such as expected radios for collection, CF, sample rate, bandwidth, and gain. The CN should then configure its attached devices based on the received parameters and begin collection. Periodically, the CN will receive updates such as the timing of the beginning of a new run or frame, or changes in collection parameters such CF or bandwidth.

Back-plane-communicated timing information should be used to label the collected data so that they may be more easily correlated with the groundtruth, and changes in collection parameters should be used to re-configure connected radios and label relevant portions of the collected data. The collected data is written and annotated in the SigMF file format. Additional information, such as detected receiver anomalies, and atmospheric conditions of temperature, dew point, and pressure could also be used to annotate the collected data.

III. EXPERIMENTAL SETUP

The BURP system is the coordination of the transmitter host and receiver hosts – CNs to create a real-world dataset. The transmitter host waits for receivers and performs a handshake to start a data collection task. The transmitter host instructs connected radios to send bursts while CNs save RF measurements using their connected receiver radios. The collected data is compared with the groundtruth in a post-processing step to create a labeled dataset of RF bursts. An overview of this process is shown in Fig. 4.

A. SCENARIO ORCHESTRATOR

The configuration for a particular type of run is specified in a JSON formatted file. Here, the user may specify the sample space for the possible data rates, CFs, modulation schemes, burst payload sizes and contents, bursts per frame, and where to save data captures. The user may choose for the run to cycle through each permutation of the specified parameters or to instead sample from these possibilities until a set number of bursts is reached. If the latter is chosen, they are expected to also specify the total number of frames.

In these configuration files, the user is also expected to supply additional information about the experiment itself. Examples of such configuration files are shown in Listing 1 and Listing 2. This information includes metadata, such as whether the conducted experiment is interior or exterior, and the specific room, and hardware configuration information about the transmitters – which antennas are connected to which transmitter, the method by which the antennas are connected (either directly or through an extension cable), and whether any have inline attenuators between itself and the antenna. The configuration file also includes run-time instructions to pass along to the receivers, such as the RX gain, bandwidth, and sample rate. This additional overhead on the part of the user is required because this information is impossible for the BURP transmitter host

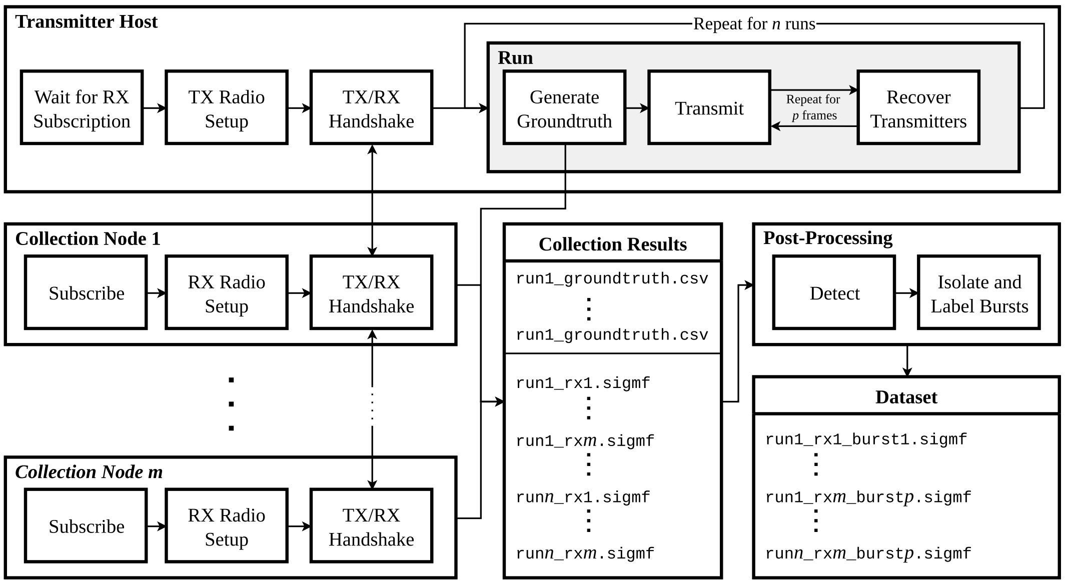

FIGURE 4. Overview of process of dataset creation with the BURP platform from start to finish.

```
{
    "data_rate": [19200, 22500],
    "tx_frequency": [390.8e6, 434.2e6],
    "tx_power": [10],
    "n_frames": 64,
    "permute_mode": "sample",
    "burst_byte_length": 255,
    "bursts_per_frame": 64,
    "contents": "random",
    "modulations": ["2fsk"],
    "dev-ant": "assume",
    "dev-ant-conn": "assume",
    "dev-attn": "assume",
    "int-ext": "INT",
    "room": "sdd_lab",
    "file_name":
 ↪  "/mnt/capture-storage/{date}_captures/
 ↪  default_{time}_{hostname}_{task_id}",
    "no_recover": false,
    "samp_rate": 2e6,
    "bw": 2e6,
    "gain": 50
}
```

LISTING 1. An example of a configuration file for the scenario orchestrator. The keyword "assume" means to use default values such as 0 for "dev-attn" (inline attenuation between transmitter and antenna), "direct" for "dev-at-conn" (transmitter-antenna connection method), or the same number as device ID for "dev-ant" (transmitter connected antenna id). Arrays, denoted by [], indicate that any of the elements may be chosen for a particular burst.

```
...
    "modulations": ["2fsk", "gfsk", "ook"],
    "dev-ant": "6-1;1-6;assume",
    "dev-ant-conn": "3-ext_cable;assume",
    "dev-attn": "7-3dB;assume",
...
```

LISTING 2. An example of possible alterations to the configuration file in Listing 1. GFSK and OOK modulation schemes are added, the antennas corresponding to devices 6 and 1 are swapped, device 3 is connected to its antenna via an extension cable, and device 7 has an attached 3dB inline attenuator.

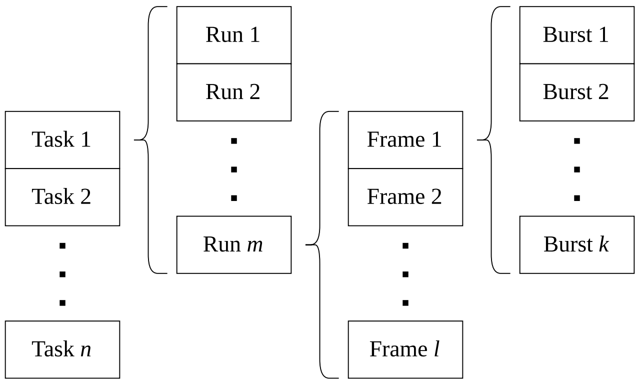

FIGURE 5. The organization structure of data creation and synchronization of transmissions.

machine to automatically determine independently. A full list of available parameters and their meanings is included in Table 1.

When a data collection is initiated, the scenario orchestrator reads every provided configuration file and generates a groundtruth for each run that conforms to the described specifications.

B. TRANSMIT SOFTWARE

The transmit software manages the radios connected to the transmitter host, publishes updates about its current status, and subscribes to updates about the status of collection nodes – the machines hosting the SDRs that are collecting measurements. Upon user prompting, the transmitter host waits for subscriptions from the specified number of collection nodes before it may begin. Once this condition is met,

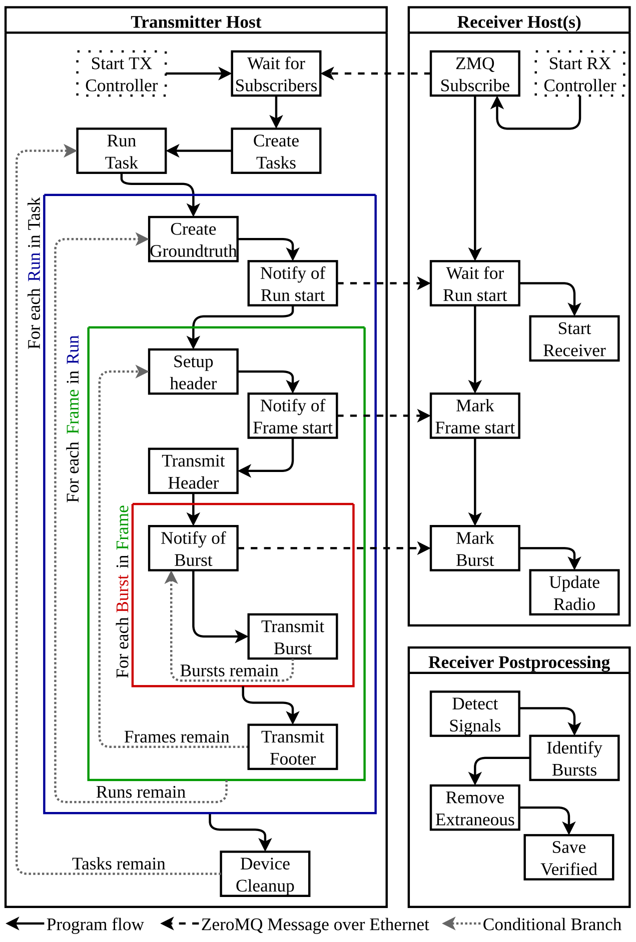

FIGURE 6. Program flow of the BURP system when operating with a transmitter host and one or more receiver host(s). A box with a dotted outline denotes an entry-point. Colors are used to denote the current rung of the synchronization structure (see Fig. 5), with blue for a run, green for a frame, and red for a burst.

TABLE 1. All available parameters for configuring a run. A single configuration file specifies the behavior of both transmitter and receiver.

Parameter	Explanation	Value(s)
`data_rate`	Possible burst bit-rates	`List(int)` baud
`tx_frequency`	Possible burst CFs	`List(int)` Hz
`tx_power`	Possible burst emitted powers	`List(float)` dBm
`n_frames`	Number of frames in a run	`int`
`permute_mode`	`"permute"` where every example in the sample space is guaranteed to occur, or `"sample"` where every example in the sample space is equally likely to occur	`"permute"` or `"sample"`
`burst_byte_length`	Length of each burst	`int` bytes
`bursts_per_frame`	Number of bursts per frame (ignored if permuting)	`int`
`contents`	Possible burst payloads. Only random is currently implemented.	`"random"`
`modulations`	Possible burst modulation schemes	Array containing any of: `"gfsk"`, `"msk"`, `"2fsk"`, `"4fsk"`, `"ook"`
`dev-ant`	Which antennas are connected to which device (mapping)	`String` map or `"assume"` for defaults
`dev-ant-conn`	How antennas are connected to each device (mapping)	`String` map or `"assume"` for defaults
`dev-attn`	Strength of inline RF attenuators for each device (mapping)	`String` map or `"assume"` for defaults
`int-ext`	Collection indoors or outdoors	`"INT"` or `"EXT"`
`room`	Collection room label	`String`
`samp_rate`	RX sample rate	`float` Hz
`bw`	RX bandwidth	`float` Hz
`gain`	RX gain	`float` dB
`no_recover`	Abort attempts to recover failed devices?	`boolean`
`file_name`	Path to save RX data	`String`

the transmitter host enqueues the specified number of tasks. Transmissions are organized where each task is a collection of runs, each run is a collection of frames, and each frame is a collection of bursts. This structure is shown in Fig. 5. The flow of operations of the transmitter host software with the receiver host software from the start to the completion of a run is shown Fig. 6.

A burst is defined as a labeled time period with distinct start and end times where an emitter is transmitting. A single frame is a sequence of bursts with a transmitted header and trailer for synchronization between the transmitting and receiving ends of the system. A header is a 300 baud 2-FSK modulated sequence of 1023 ones, 1 zero, the unique frame ID, then 1 zero and 1023 ones. The baud rate is set to 300 to reduce the probability of errors when detecting the final bit. Since the header and footer are intended to be pure tones, the baud rate is irrelevant during the majority of the signal. This, in effect, creates a pure-tone marker that can be used to cross-reference between receivers the exact time when frames begin.

The received signal for an example frame in the frequency domain is shown in Fig. 7. The frame header, frame footer, and 16 bursts sandwiched between them are visible and annotated. The same sort of frame, instead with 6 bursts, is shown from the perspective of a receiver in the time domain in Fig. 8. The exact CF of this signal is dependent on the run configuration file.

The same structure of signal for the header is sent for the trailer, with the beginning and ending tones carrying inverted data as compared to the header, so the marker tone is at a different frequency than the header. A run corresponds to a singular ‘use’ of a configuration file from start to finish, so a task may therefore indicate the specification of multiple uses of a configuration file.

The transmitter host publishes that a run is beginning, and waits for subscribers to send a ‘ready’ signal. The software now begins the ‘radio setup’ phase, where it logs all connected transmitters and performs a recovery sequence if any faults are detected. Then, it uses the scenario orchestrator to generate a groundtruth sequence for this run and begins the first frame, transmitting the frame header. Within the frame, the transmitter host conducts individual bursts, where the parameters of each burst are published so that collected data may be annotated in real-time by the receivers. Once

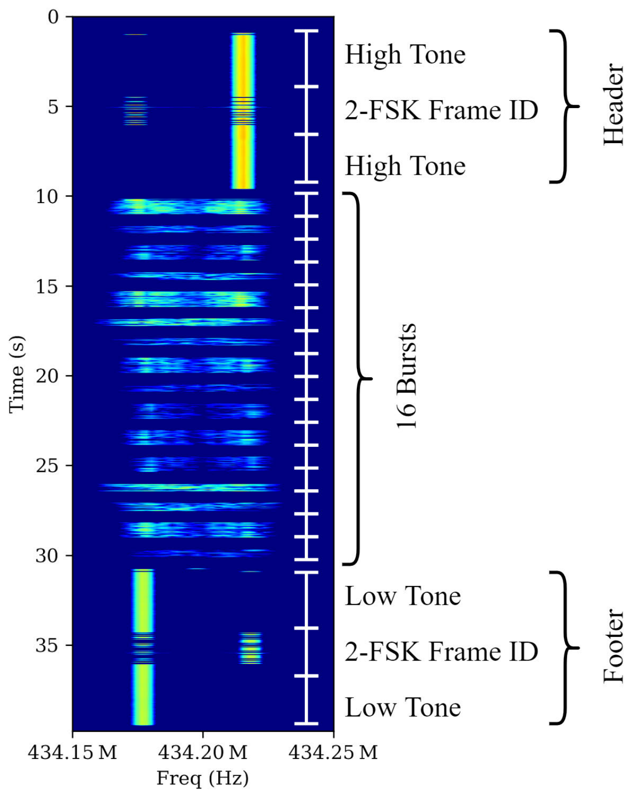

FIGURE 7. Annotated spectrogram of example frame with respective headers, sequence of bursts, and then footers.

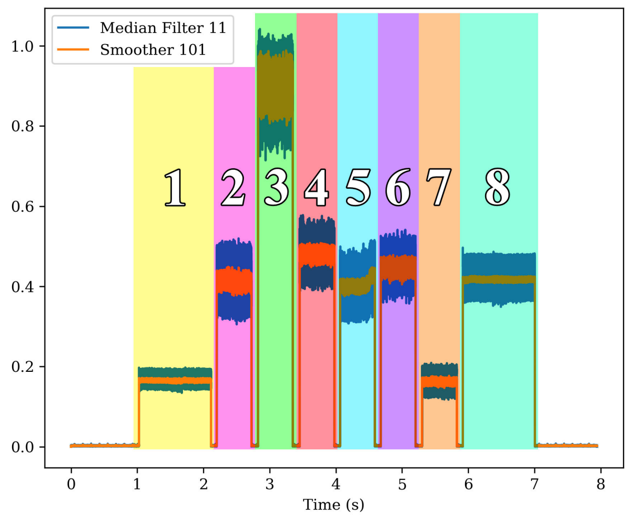

FIGURE 8. Example power vs. time of 6-burst example frame. The y-axis represents relative received power (uncalibrated). Segments 1 (yellow) and 8 (cyan) contain the header and footer, respectively. Segments 2-7 contain individual bursts with data modulated in 2-FSK.

all bursts in a frame are complete, the trailer is transmitted to indicate the end of a frame. Within a frame, the software may detect that some transmitters are unresponsive or not working properly. Between frames, the software cycles power to faulty devices to automatically recover them so the run may continue as normal. When all frames have been transmitted, the current run ends and the next run begins. This cycle repeats until all tasks are completed and the transmitter software exits.

C. TRANSMIT HARDWARE

The BURP transmitter host is a specially constructed PC to support the maximum possible number of USB devices at once on a single bus. A diagram of how the significant components of the system are connected is shown in Fig. 11 and photographs of the assembled machine are shown in Fig. 9 and Fig. 10. The transmitter host machine uses an Intel i7-12700 CPU with 64GiB DDR4 RAM @ 4800MHz in a ASUS Prime Z690-P motherboard with Intel Z690 chipset with a 14TB HDD and 2× 2TB Samsung 970 Evo SSDs in RAID 1 configuration. A tree of internal USB hubs, 3× 4-port hubs connected to a 4-port hub that is connected to the motherboard, support an array of 12 externally-facing drive-bay-mounted 10-port USB-A hubs, culminating in a total of 120 externally-facing USB-A ports on the front panel alone.

Each of the externally-facing hubs is “dumb” in that it contains no internal memory, serial number, or otherwise identifying information. An indexing flash storage drive with a unique serial number is connected to each hub to allow the system to ascertain a certain USB “coordinate system,” where each device connected to the system may have its logical location as the operating system sees it correlated to a physical location on the front panel. This information is used within the transmit software to label data within the groundtruth so that possible patterns of received power level or multi-path effects may be later accounted and controlled for by any user of the generated data. It is possible to index the system once, in a setup stage, then remove the indexing flash drives to make space for additional radio devices. Testing has shown, however, that the USB coordinate system will occasionally and unpredictably re-arrange itself, and therefore this process must be done after each full power cycle of the motherboard.

Since all of the YS1s are connected simultaneously through USB, it is possible to give instructions to multiple devices for overlapping co-channel or adjacent channel transmissions where more than one radio is transmitting at a time. This can be used to generate more varied datasets and train SEI or AMC models in less predictable or favorable conditions. The transmitter host also has the portability of a typical desktop computer, meaning it can be relocated and set up in different areas of the same room to achieve variable harmonic and multi-path effects.

The system has a custom-built power delivery system with two power supplies; a diagram of this arrangement is shown in Fig. 11. There is a 1000W main power supply for the motherboard, CPU, storage, additional sensors, and typical peripherals, and a 300W auxiliary power supply for the exterior USB hubs. This auxiliary power supply is required because the 5V rail of the main power supply is unable to supply enough current to exterior USB hubs

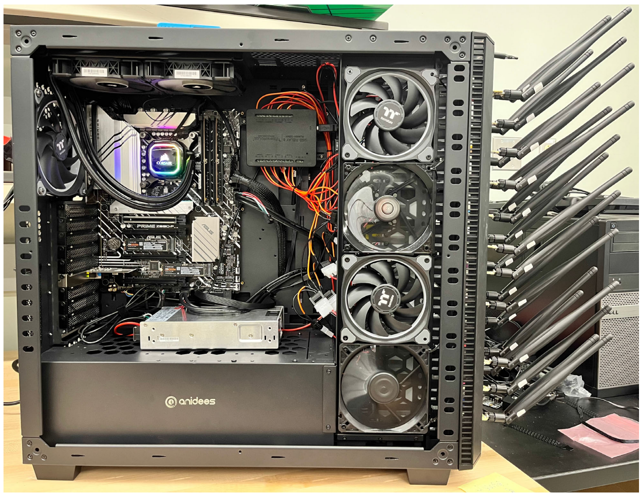
FIGURE 9. Side profile of the transmitter host machine with the side panel removed. The motherboard occupies the back of the case and the main PSU (not visible) is in the bottom compartment. On top of the bottom compartment is the auxiliary PSU, which is connected to the USB hub relays near the top-front of the case. The tip of the atmospheric probe is visible on the top front of the case above the bank of antennas.

when more than a couple hubs are fully populated with transmit devices. The power distribution from the auxiliary power supply is shown in Fig. 12. The power distribution to the 12 exterior USB hubs is routed through a Numato Lab 8 Channel USB solid-state relay module [52], which can connect and disconnect power to the USB hubs depending on software control. Routing power through relays allows the transmit software to attempt to automatically recover misbehaving or unresponsive devices by simulating removing and re-inserting the device through a power cycle of the hub the device is hosted on without the associated mechanical wear or required operator intervention of physical re-insertion.

Also connected to the transmitter host is a Dracal PTH-200 USB temperature, humidity, and atmospheric sensor [53]. This sensor allows the transmit software to measure and log temperature, dew point, and atmospheric pressure, which are used to label each data frame.

D. RECEIVE HARDWARE

Each of the receive-end CNs are ITX form-factor computers with an Intel i5-11600K processor, 64GiB RAM, 2TB NVMe solid-state drive, and 14TB 7200 RPM hard disk drive, along with a PCIe expansion card for 10GbE SFP+ networking. The SFP+ network expansion card is necessary for collections using certain high-data-rate SDRs such as an Ettus Research USRP X310 or N320. Each CN can host a single SDR receiver and is re-configurable to accept any radio that can communicate with GNURadio through USB or Ethernet through SFP (up to 10 GbE) or RJ45/8P8C (up to 1 GbE). For the initial proof-of-concept model, the system was designed to support the Ettus Research USRP B210 and E310 and SignalHound SM200C.

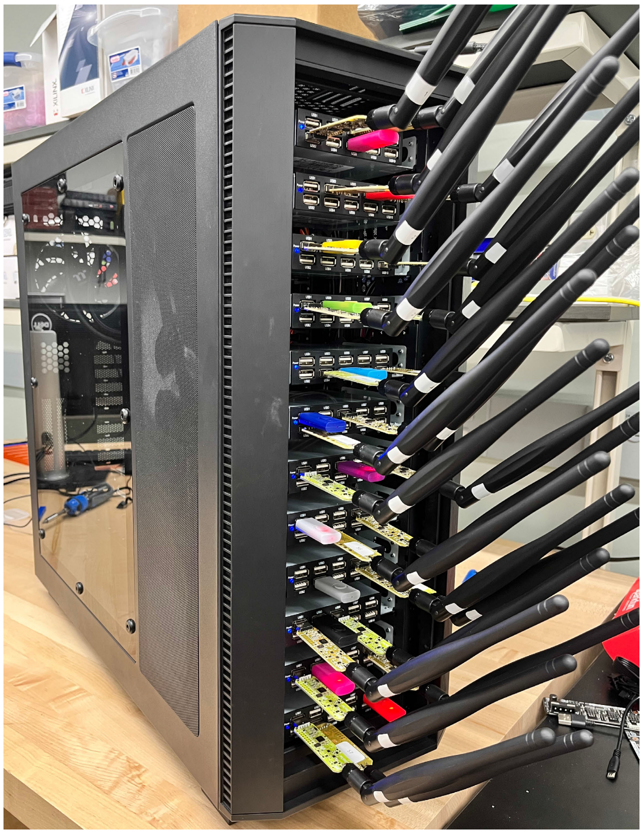
FIGURE 10. Front bank of USB hubs on the transmitter host machine with 20 Yard Stick One devices distributed throughout.

E. RECEIVE SOFTWARE

The receive software subscribes over ZeroMQ to an ongoing session of the transmit software. Each of the CNs manage their own connected radio receiver and react to notifications received from the transmit software through the control backplane. When the receive software receives a notification that a run is starting, the connected radio receiver is turned on and then a "ready" message is sent back to the transmitter host. While the system is running, the CNs monitor and re-configure their connected radio on-the-fly. When a notification for a frame barrier or a burst is received, the CN instructs the connected radio to adjust the CF to best capture this data. Since the data is output in the SigMF file format, these changes are captured in annotations. When the run finishes, the radio is switched back off.

Post-processing is split into two primary steps, as shown in Fig. 13: signal detection and burst isolation/labeling. Both are entirely implemented in Python.

1) SIGNAL DETECTION

The signal detection stage uses a simple energy detection algorithm to identify the location (i.e. start/stop indices) of each burst within the provided SigMF data file, and does not distinguish between header, footer, and bursts. Detection is performed as follows: after loading the complex IQ data from the SigMF data file and calculating the magnitude of each

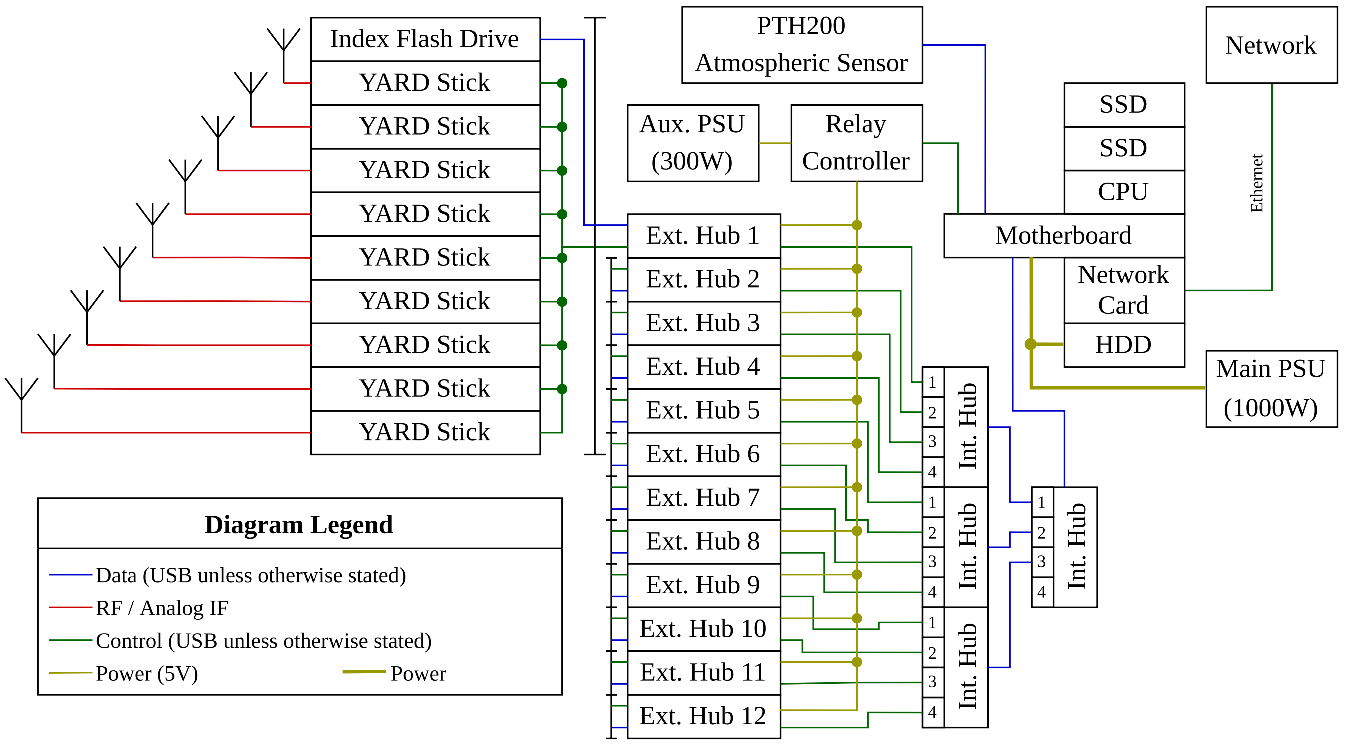

FIGURE 11. Data and control structure for BURP system.

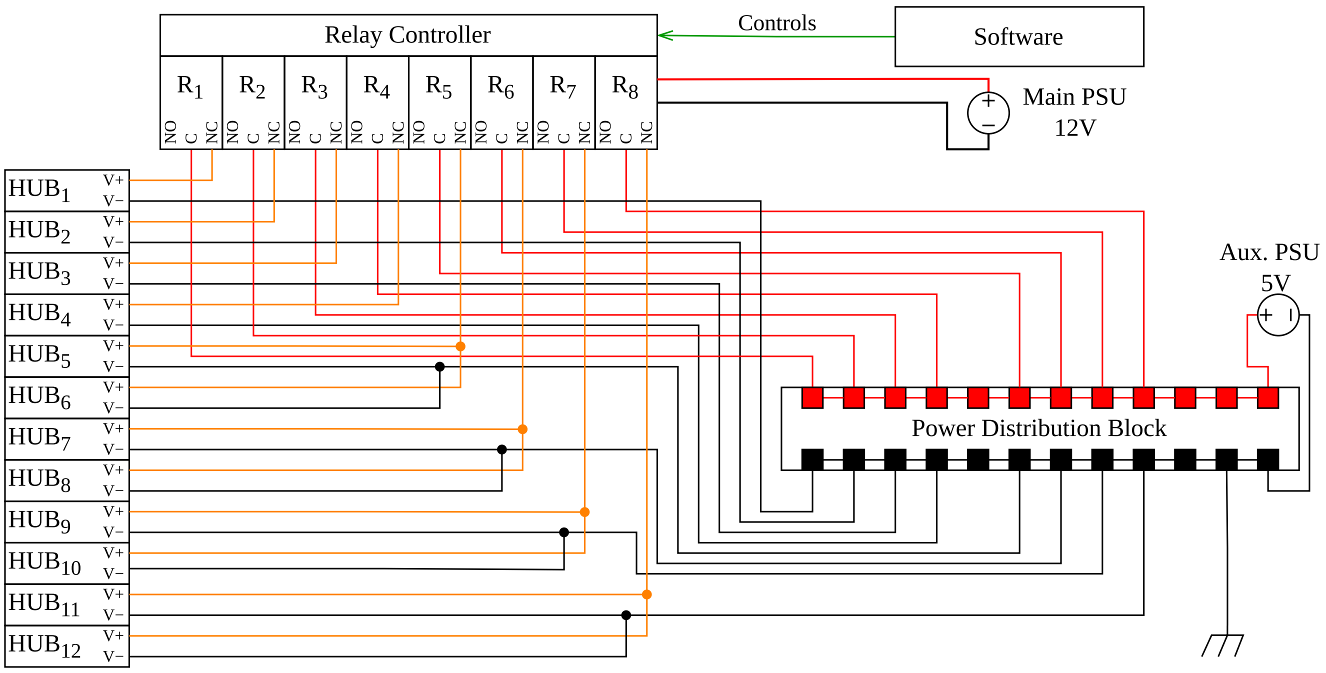

FIGURE 12. Power distribution diagram showing relay control of USB hub power.

sample, we perform iterative smoothing of the magnitude array. The magnitude of the noise floor is estimated from the first n samples of the smoothed magnitude, where n is a tunable hyperparameter. Then, looping over the smoothed magnitude, when the value of the smoothed magnitude is greater than the noise floor plus some threshold (a tunable parameter), we denote the start of a burst. When the value of the smoothed magnitude returns below the noise floor plus threshold for at least m samples (a tunable parameter), we denote the end of the burst. Any extra or missed bursts

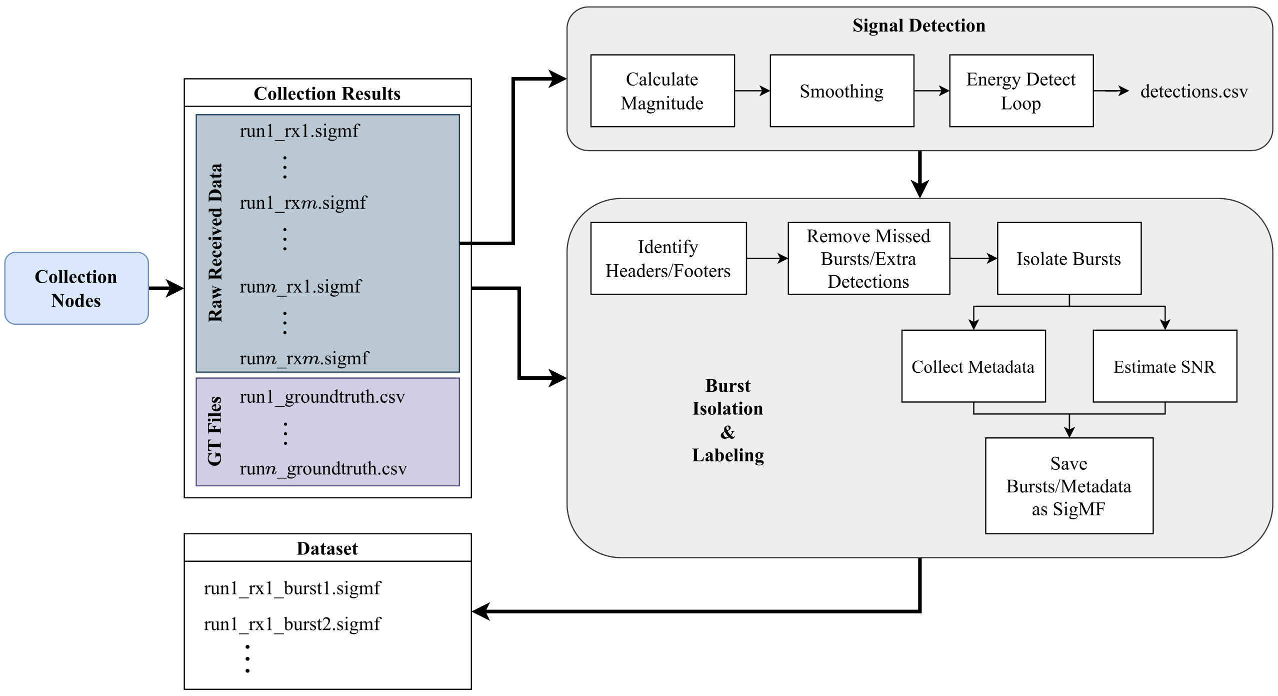

FIGURE 13. An overview of the BURP post-processing stages.

detected are removed during the second stage of post-processing: burst isolation and labeling.

2) BURST ISOLATION & LABELING

The burst isolation and labelling stage uses the output of the previous signal detection stage, along with the SigMF data, metadata, and BURP groundtruth files to save each individual burst in separate SigMF data files with associated metadata containing the timestamp, estimated signal-to-noise ratio (SNR), estimated CF offset (CFO), modulation scheme, transmitter/receiver ID, temperature, dew point, and atmospheric pressure measurements, and all other parameters denoted in the configuration file. To prevent exceeding available memory, we process up to 10^6 samples at a time, each in a separate process, with optional multi-threading. First, header/footer bursts are identified using the groundtruth timestamps, and the location of each non-header/footer burst is verified by extrapolating from the header/footer timestamps using the sample rate. Any extra or missed bursts detected in the previous stage are identified through a mismatch in groundtruth timestamp and extrapolated time, and are removed from the detections or groundtruth lists. All headers and footers are also removed from the detections list, so as not to be included in the dataset. Then, for each verified burst in the detections list, the contents of the burst is saved to a SigMF data file with the name 'runID_frameID_timestamp.sigmf-data'. All other available metadata (i.e. modulation scheme, transmitter/receiver ID, temperature/humidity) are collected from the groundtruth files and saved in the associated 'runID_frameID_timestamp.sigmf-meta' file along with the estimated SNR of the burst, which is calculated using the noise floor before and after the burst, and the estimated CFO.

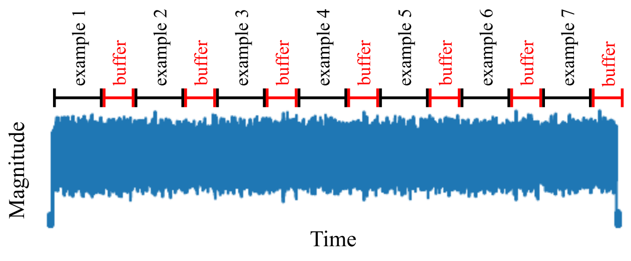

FIGURE 14. Each burst can be ingested and split into segments or examples of desired length within the ML framework of choice (PyTorch, Tensorflow, etc). To ensure each example is distinct from one another, we encourage placing a buffer between each example taken from the data file.

For RFML model training/evaluation, each burst can be filtered by metadata parameters, ingested, and split into segments of desired length within a framework of choice (PyTorch, Tensorflow, etc). For example, we can select only bursts transmitted by device 37, received by CN 2, or transmitted and collected at 434.2MHz, and split these bursts into examples of 1024 IQ samples. To ensure each training example is distinct from one another, we encourage placing a buffer between each example taken from the data file, as shown in Fig. 14.

IV. TESTING AND VERIFICATION

The primary goal of the verification process was to determine if the datasets produced by BURP adhered to the specifications in the configuration files. For this, transmission parameters were set to specific values and the measured outputs were compared to the desired results. In the testing process, the goal was to determine the overall stability and performance of the system in terms of realized data-generation capabilities.

A. VALIDATION OF OUTPUTS

It is critically important that data is representative of the desired learned behaviors for ML applications. Future inference performance will only meet expectations if training data is similar to the intended deployment environment. As previously discussed, sets of bursts, over-the-air transmissions, are structured into frames and parsed to be inputs for a chosen NN structure. When parsing, data validation included verification that (1) energy on the expected frequency is detected well above the noise floor and formed into a distinct envelope with a defined start time and end time, (2) timestamps and time-duration of each burst aligned with that recorded in the groundtruth files, (3) no YS1 emitters from a previous burst within the same frame are stuck ON and continuing to transmit during the same time period, and (4) no other observable anomalous conditions are present.

These validation steps were performed using traditional signal processing techniques, assuming the truth of the recorded groundtruth files. A future extension of this validation process, not yet fully implemented in the current iteration, is to use a dedicated RX node in Fig. 2 as a co-located "point blank" local monitor of the generated signals for a high fidelity, high SNR capture without most channel perturbations. This would also enable a real-time closed feedback loop where the BURP transmitter host would be able to detect and mitigate the situation where a YS1 emitter is continuing to transmit past the expected end time of an assigned burst – currently a source of degradation in collected data.

B. STABILITY AND PERFORMANCE

The BURP framework required substantial debugging effort to achieve nominal operations. Most of the challenges revolved around the unpredictability of the chosen low-cost YS1 emitters and practical limitations of the number of xHCI USB endpoints available with the Intel Z690 chipset. Nevertheless, the initial iteration of BURP has proven to be capable of generating real-world datasets of sufficient size for medium-scale experimentation for both AMC and SEI at a faster rate than most existing capabilities. Future planned incarnations of the platform are also discussed, addressing identified limitations.

1) DATA GENERATION RATE

To achieve distinct envelopes around transmissions that can be isolated in time, each burst is conducted in serial with a brief wait period in-between. Taking into account various delays and fault-recovery procedures, the theoretical rate of data generation for the initial revision of BURP is a relatively slow 1,380 bursts per hour. Given a theoretical target SEI dataset at just one location with 1M bursts (100 devices with 10k bursts recorded per device) it would take this configuration an estimated 724 hours (30 days) of continuous operation to complete. Given a theoretical target AMC dataset at just one location with 50k bursts (5 modulation schemes with 10k bursts recorded per modulation scheme), it would take this configuration an estimated 36 hours (1.5 days) of continuous operation to complete.

The rate of data generation could be further increased for a single transmitter host with the substitution of higher-quality and more capable radios. Higher quality radios could be used to provide near continuous timestamped bursts since the precision of the recorded start and stop times would be increased considerably, reducing the required wait time to aid in signal processing (t_{delay}). This would also reduce the time required to recover failed devices between frames to nearly zero, since most radios rarely fail as frequently as the chosen YS1s. Anticipated upgrades with better radios with reliability comparable to USRPs, minimized inter-burst latencies, and payload symbol rate increased to 500k baud would result in a 10x speed-up for a theoretical data generation rate of 13,900 bursts per hour. With these anticipated upgrades, to complete the aforementioned theoretical SEI and AMC datasets it would take 72 hours (3 days) and 3.6 hours, respectively.

Experimentally, during continuous operation, the data generation rate of the initial iteration of BURP was measured to be closer to 1,300 bursts per hour. This rate does not hold constant, but decreases over time as the rate of failures increases and overall performance decreases (See: Section IV-B2). Furthermore, continuous operation must be occasionally interrupted to copy collected data from the collection nodes to another location where analysis is performed.

The theoretical rate of data generation (r_{tx}) is calculated as follows, where each symbol and its meaning is described in Table 2:

$$t_{burst} = t_{config} + t_{wait} + \frac{l_{burst}}{f_{burst}} \tag{1}$$

$$t_{header} = t_{footer} = t_{config} + \frac{2 \times l_{flag} + 4 \times l_{id}}{f_{flag}} \tag{2}$$

$$\begin{aligned} t_{frame} = t_{gt} + t_{header} + n_{bursts}(t_{burst} + t_{delay}) \\ + t_{footer} + t_{recover} \end{aligned} \tag{3}$$

$$t_{run} = t_{setup} + (n_{frames} \times t_{frame}) \tag{4}$$

$$r_{tx} = \frac{n_{bursts} \times n_{frames}}{t_{run}} \tag{5}$$

2) STABILITY AND PERFORMANCE DECAY

The emitters connected to the transmitter host are prone to random and unpredictable failures, that may be categorized into two types: "bad transmissions," where a dropped packet when issuing instructions to the radio causes the radio to repeatedly and infinitely transmit the contents of its buffer instead of the intended message, and "unresponsive radio," where the radio stops responding to messages from the host

TABLE 2. Definitions of and values for variables used in calculating the theoretical continuous data generation rate of BURP. Entries whose values are in italics are derived from other values in the table according to the described relationships. Entries whose values are in bold are parameters set by the user before a run. All other values of entries are either hard-coded or observed properties of the system.

Symbol	Meaning	Value
t_{burst}	Time required to transmit one burst	*0.43s*
t_{header}	Time required to transmit a header	*1.17s*
t_{footer}	Time required to transmit a footer	*1.17s*
$t_{recover}$	Avg. time required to recover all devices	*40s*
t_{frame}	Time required to execute an entire frame	*2m 46s*
t_{run}	Time required to execute an entire run	*2h 58m*
t_{config}	Time required to configure a YS1	0.3s
t_{wait}	Wait time between configuration and TX	0.1s
t_{delay}	Wait time to aid in signal processing	1.5s
t_{gt}	Time to generate groundtruth for a frame	0.25s
t_{setup}	Time required to set up a run	20s
l_{flag}	Size of pure-tone frame marker	128 bytes
l_{id}	Size of frame unique id	16 bytes
l_{burst}	Size of burst payload	**1,024 bytes**
f_{flag}	Symbol rate of frame markers	300 baud
f_{burst}	Symbol rate of burst TXs	**31,250 baud**
n_{bursts}	Number of bursts in a frame	**64**
n_{frames}	Number of frames in a run	**64**

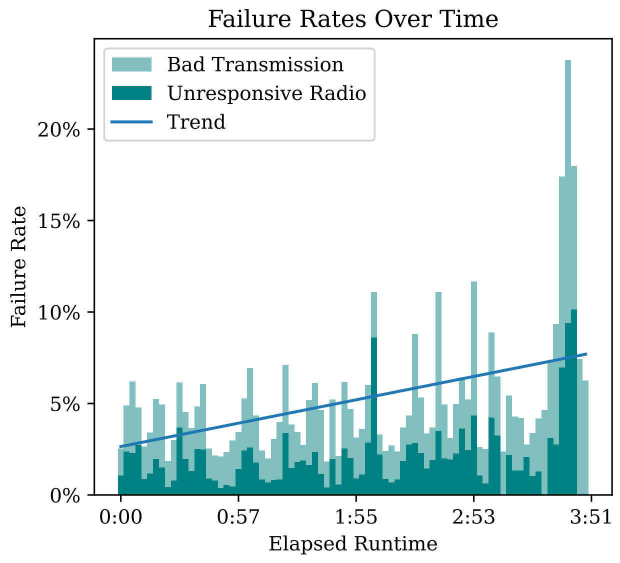

FIGURE 15. Encountered failures over the course of normal data-collection runs. Each bin represents the ratio of failures to total attempted bursts over a 2m 53s time period. The trend line is the line of best fit in the least-square sense.

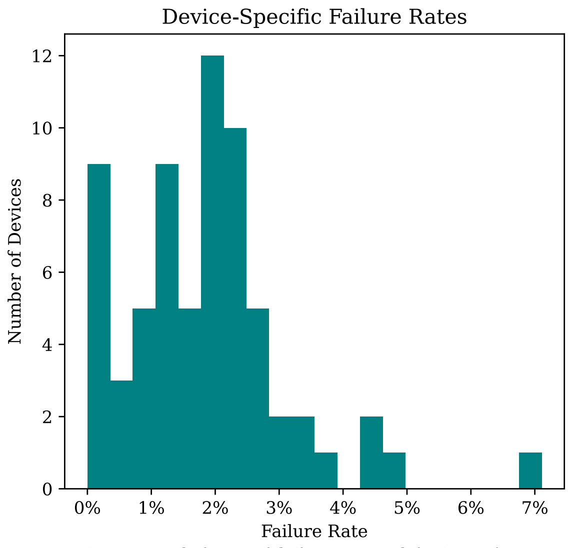

FIGURE 16. Histogram of observed failure rates of devices. The distribution has a mode around 2% and superficially resembles a normal distribution, but with clipping at 0% and several "lemon" devices not included in this figure that have a greater than 99% failure rate.

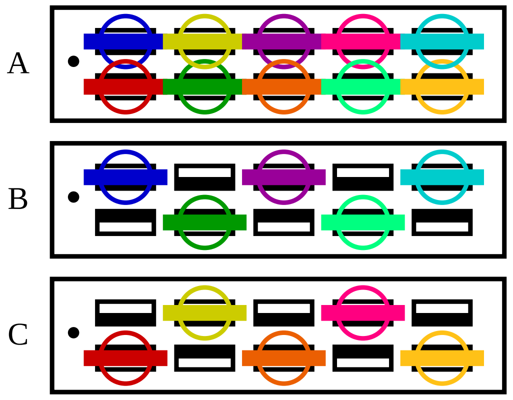

FIGURE 17. Three different USB hub population patterns of transmitters. Each colored rectangle shows the footprint of each YS1 PCB when viewed from the front and each colored circle shows the footprint of the connected antenna. Pattern A is full-density and has "crushing" while Patterns B and C are half-density and do not have crushing.

system, causing USB timeout errors. The trend is captured in Fig. 15, where the number of anomalies attributable to these two causes are shown to generally increase over time, leading to a decrease in overall system stability as time goes on. The failure rate across the population of devices varies, as shown in Fig. 16, with some devices having near-perfect reliability and others having an anomaly arise in over 7% of the attempted transmissions.

C. LIMITATIONS

1) HARDWARE CONSIDERATIONS

While technically possible to fully populate the USB array with YS1 transmitters as shown in Fig. 17-A, the connected antennas have vertical footprints that are too large to stack neatly and the YS1s are "crushed" askew. This crushing places strain on both the connectors of the YS1s and the USB ports, causing damage over time to the components. Instead, to preserve the devices and USB ports, the devices were placed in a "W" or "M" pattern as shown in Fig. 17-B and Fig. 17-C, respectively. This reduces the number of supported devices with 10 USB hubs from 100 to 50.

When connected through a USB interface, a single YS1 was observed to draw, at a steady-state, 9 mA (45 mW) at idle. Active power draw varies depending on CF and modulation scheme. The highest power draw at steady-state while actively transmitting was measured to be 61 mA (305 mW) when the modulation scheme was 2-FSK with a CF between 429 MHz and 444MHz. Additionally, the marker flash drives were observed to draw a steady-state 41 mA (205 mW).

Therefore, the fully populated BURP transmitter host USB hub array with 1 active transmitter, 89 idle transmitters, and

10 marker flash drives draws around 1.3 A (6.3 W) from connected devices alone. The total power draw is increased further by the connected relays, USB hubs, and parasitic resistances. While this power draw seems relatively small, it was still enough to overwhelm the 5 V rail of the main PSU, a 1000 W consumer PC power supply. When connected to a typical PC PSU, the turn-on transient power consumption of this many connected devices trips protection circuitry and causes power to be cut to the entire system. If individual relays are turned on gradually, the 5 V rail of the main PSU is still unable to supply enough power in the steady-state and the YS1 behavior becomes significantly more erratic and less reliable due to brown-outs. To mitigate this, we chose to include an auxiliary 5 V power supply as discussed previously in Section III-C.

2) TRANSMITTER LIMITATIONS

The chosen transmitter, the Yard Stick One, is not an SDR as traditionally defined, and is instead marketed as a ''wireless test tool,'' and this paper describes using them in ways never intended by the manufacturer [49]. The firmware, RfCat, is open-source and is available online [54]. The firmware required modifications for flashing unique serial numbers, device IDs, to each device and to allow addressing by this serial number for a large number of connected devices.

The YS1 is limited to the transmitting within the 300-348 MHz, 391-464 MHz, and 782-928 MHz frequency bands, with ASK/OOK, GFSK, 2-FSK, 4-FSK, and MSK modulation schemes, at power levels from -30 to +10 dBm. The tool can transmit up to 255 bytes at 500k baud, but was found to be limited to an upper limit of around 230.4k baud for payloads of up to 65535 bytes. The device is unable to transmit messages larger than 65535 bytes. Only one transmitter may be ''active'' at a time when connected to the same host and the system is incapable of utilizing more than one YS1 simultaneously, (e.g., multiple threads each managing one YS1). When operating within specifications, the device is still prone to random and unpredictable failures; these manifest as a dropped packet when communicating with the device, causing the device to continue transmitting the last received packets indefinitely, or the device outright refusing to respond to communication, causing USB timeout errors. When used in its intended role, these issues are minor annoyances, but when used to generate large amounts of data as part of an automated testbed, they compound to become a major design consideration, lest the system become completely unreliable.

D. SAMPLE DATASET

The sample dataset includes transmissions from 30 YARD Stick One emitters at multiple CFs (346.3 MHz, 416.4 MHz, 783.7 MHz). Each run, repeated 3 times at different CFs, included 64 frames of 64 bursts, each of which had a payload of 1024 bytes of randomized data transmitted at 31,250 baud. The 3 co-located receivers were configured with a 250 kHz sample rate and 250 kHz bandwidth with a gain of 50 dB. Each radio, emitter and receiver, was connected to an L-com 900 MHz 3 dBi rubber duck antenna which remained the same throughout the course of the collections. Each set of 3 runs at each CF used randomized emitter positions across the transmitter host USB hub array. At this baud rate and payload length with this sample rate, each capture of a burst is 65,536 samples per receiver that may be fed into the front-end of an SEI or AMC model.

The data was collected at three locations as follows:

- TX and RXs in the same room, direct line of sight: ~150k bursts (~29.5G samples)
- TX and RXs in the same room, metal furniture obstructing line of sight: ~150k bursts (~29.5G samples)
- TX and RXs in different rooms, separated by wall: ~150k bursts (~29.5G samples)

The sample dataset is forthcoming and will be published for other researchers after we validate it in ongoing SEI and AMC experiments. We have already performed preliminary training of SEI and AMC models using the data to indicate the quality of the generation process. We intend to make a subsequent review of BURP to increase stability and reliability before eventually publishing datasets as large as permitted by IEEE DataPort.

E. FUTURE WORK

The core goal of the BURP framework is to support data generation for a wide variety of RFML experiments where previously only simulated data was available. To maximize the rate of data generation and minimize downtime of the platform, its performance and reliability are the subject of ongoing development. One such measure to increase reliability is the addition of a co-located receiver on the transmitter host able to provide measurements as a local monitor of emitted signals for high-fidelity, high-SNR captures before most channel and multipath effects occur. In addition, a local monitor would provide an important role in self-correction of transmission hardware and software anomalies that tend to occur at run time.

The possibilities of extensions to the platform for increased capability include modifications to support:

- Heterogeneous emitters
- Higher or lower quality receiver and transmitter SDRs
- Non-communication emitters (RADAR, etc.)
- Multiple, dispersed, transmitter hosts
- More receiver hosts
- Modular transmitter hosts
- Weatherproofed outdoor transmitter and receiver hosts
- Miniaturized transmitters hosted on commercial drones

Switching to higher quality transmitter SDRs would especially improve the resolution of groundtruth labels to provide near continuous timestamped bursts. The inclusion of more than one transmitter host in the system would also require additional infrastructure in the form of a central management server and monitor to coordinate all nodes in the system.

On top of hardware modifications, the rate of useful dataset generation may be further increased through the addition of a dataset augmentation step. The merits and limits of this process are discussed in a previous work [27], where it is shown that augmentation can drive a performance increase in trained AMC algorithms. Augmentation for SEI algorithm training is expected to be limited unless extremely high fidelity RF transmit chain models are integrated.

The platform may be used to generate data for experiments to better understand and account for real-world conditions whose effect on RFML algorithms are not currently well understood, such as interference, multipath, fading, and thermal effects. Assessing the effect of factors on SEI and AMC performance are the subject of future publications.

V. CONCLUSION

This approach to generating well-labeled real-world RF datasets is cost-effective for the amount of high-quality data that is produced and is easily extensible. The initial sample use-case focused on just one type of emitter and just a few modulation schemes, but the system is easily augmented to be conducive to a large number of RFML applications. The current system can scale to an arbitrary number of receiver nodes, but will require additional infrastructure to be developed to scale to multiple transmitter node cases. This could then potentially be extended to a point where 100 transmitter and receiver nodes with diverse emitters and receivers are present within a system allowing the development of very generalized RFML models.

REFERENCES

[1] T. O'Shea and N. West, "Radio machine learning dataset generation with GNU radio," in *Proc. GNU Radio Conf.*, 2016, vol. 1, no. 1, pp. 1–6.

[2] R. Elbakly, H. Aly, and M. Youssef, "TrueStory: Accurate and robust RF-based floor estimation for challenging indoor environments," *IEEE Sensors J.*, vol. 18, no. 24, pp. 10115–10124, Dec. 2018, doi: 10.1109/JSEN.2018.2872827.

[3] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.

[4] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1938–1948, Apr. 2019, doi: 10.1109/JIOT.2018.2872441.

[5] N. West, T. O'Shea, and T. Roy, "A wideband signal recognition dataset," in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Lucca, Italy, Sep. 2021, pp. 6–10, doi: 10.1109/SPAWC51858.2021.9593265.

[6] W. H. Clark and A. J. Michaels, "Quantifying dataset quality in radio frequency machine learning," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, San Diego, CA, USA, Nov. 2021, pp. 384–389, doi: 10.1109/MILCOM52596.2021.9652987.

[7] J. M. McGinthy, L. J. Wong, and A. J. Michaels, "Groundwork for neural network-based specific emitter identification authentication for IoT," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6429–6440, Aug. 2019, doi: 10.1109/JIOT.2019.2908759.

[8] L. J. Wong, W. H. Clark IV, B. Flowers, R. M. Buehrer, A. J. Michaels, and W. C. Headley, "The RFML ecosystem: A look at the unique challenges of applying deep learning to radio frequency applications," 2020, *arXiv:2010.00432*.

[9] J. Robinson, S. Kuzdeba, J. Stankowicz, and J. M. Carmack, "Dilated causal convolutional model for RF fingerprinting," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, Jan. 2020, pp. 0157–0162, doi: 10.1109/CCWC47524.2020.9031257.

[10] Y. Lin, J. Jia, S. Wang, B. Ge, and S. Mao, "Wireless device identification based on radio frequency fingerprint features," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6, doi: 10.1109/ICC40277.2020.9149226.

[11] L. Ding, S. Wang, F. Wang, and W. Zhang, "Specific emitter identification via convolutional neural networks," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2591–2594, Dec. 2018, doi: 10.1109/LCOMM.2018.2871465.

[12] G. Huang, Y. Yuan, X. Wang, and Z. Huang, "Specific emitter identification based on nonlinear dynamical characteristics," *Can. J. Electr. Comput. Eng.*, vol. 39, no. 1, pp. 34–41, Winter. 2016, doi: 10.1109/CJECE.2015.2496143.

[13] J. Gong, X. Xu, and Y. Lei, "Unsupervised specific emitter identification method using radio-frequency fingerprint embedded InfoGAN," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2898–2913, 2020, doi: 10.1109/TIFS.2020.2978620.

[14] A. Aubry, A. Bazzoni, V. Carotenuto, A. De Maio, and P. Failla, "Cumulants-based radar specific emitter identification," in *Proc. IEEE Int. Workshop Inf. Forensics Secur.*, Iguacu Falls, Brazil, Nov. 2011, pp. 1–6.

[15] L. J. Wong, "On the use of convolutional neural networks for specific emitter identification," MSEE, Virginia Tech., Blacksburg, VA, USA, Tech. Rep. 15412, Apr. 2018. [Online]. Available: https://vtechworks.lib.vt.edu/bitstream/handle/10919/83532/WongLJT2018.pdf

[16] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, San Francisco, CA, USA, Sep. 2008, p. 116, doi: 10.1145/1409944.1409959.

[17] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, 1st Quart., 2016, doi: 10.1109/COMST.2015.2476338.

[18] J. Hall, M. Barbeau, and E. Kranakis, "Enhancing intrusion detection in wireless networks using radio frequency fingerprinting," in *Proc. Commun., Internet, Inf. Technol.* St. Thomas, US Virgin Islands: ACTA Press, 2004, pp. 201–206.

[19] L. J. Wong, W. C. Headley, S. Andrews, R. M. Gerdes, and A. J. Michaels, "Clustering learned CNN features from raw I/Q data for emitter identification," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Los Angeles, CA, USA, Oct. 2018, pp. 26–33, doi: 10.1109/MILCOM.2018.8599847.

[20] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep learning convolutional neural networks for radio identification," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 146–152, Sep. 2018, doi: 10.1109/MCOM.2018.1800153.

[21] X. Li, F. Dong, S. Zhang, and W. Guo, "A survey on deep learning techniques in wireless signal recognition," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–12, Feb. 2019, doi: 10.1155/2019/5629572.

[22] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Chicago, IL, USA, Jun. 2010, pp. 383–392, doi: 10.1109/DSN.2010.5544294.

[23] S. Guo, R. E. White, and M. Low, "A comparison study of radar emitter identification based on signal transients," in *Proc. IEEE Radar Conf.*, Oklahoma City, OK, USA, Apr. 2018, pp. 0286–0291, doi: 10.1109/RADAR.2018.8378572.

[24] S. Apfeld and A. Charlish, "Recognition of unknown radar emitters with machine learning," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 6, pp. 4433–4447, Dec. 2021, doi: 10.1109/TAES.2021.3098125.

[25] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "ORACLE: Optimized radio clAssification through convolutional neuraL nEtworks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 370–378.

[26] K. Chowdhury, S. Ioannidis, and T. Melodia, "Deep learning for RF signal classification and fingerprinting," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Mar. 2019, pp. 598–603.

[27] W. H. Clark, S. Hauser, W. C. Headley, and A. J. Michaels, "Training data augmentation for deep learning radio frequency systems," *J. Defense Model. Simul., Appl., Methodol., Technol.*, vol. 18, no. 3, pp. 217–237, Jul. 2021, doi: 10.1177/1548512921991245.

[28] P. Wang and M. Vindiola, "Data augmentation for blind signal classification," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 305–310, doi: 10.1109/MILCOM47813.2019.9020842.

[29] N. Soltani, K. Sankhe, J. Dy, S. Ioannidis, and K. Chowdhury, "More is better: Data augmentation for channel-resilient RF fingerprinting," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 66–72, Oct. 2020.

[30] C. Comert, M. Kulhandjian, O. M. Gul, A. Touazi, C. Ellement, B. Kantarci, and C. D'Amours, "Analysis of augmentation methods for RF fingerprinting under impaired channels," in *Proc. ACM Workshop Wireless Secur. Mach. Learn.*, May 2022, pp. 3–8, doi: 10.1145/3522783.3529518.

[31] A. Al-Shawabka, P. Pietraski, S. B. Pattar, F. Restuccia, and T. Melodia, "DeepLoRa: Fingerprinting LoRa devices at scale through deep learning and data augmentation," in *Proc. 22nd Int. Symp. Theory, Algorithmic Found., Protocol Design Mobile Netw. Mobile Comput.*, Jul. 2021, pp. 251–260, doi: 10.1145/3466772.3467054.

[32] N. M. Müller and K. Markert, "Identifying mislabeled instances in classification datasets," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Budapest, Hungary, Jul. 2019, pp. 1–8, doi: 10.1109/IJCNN.2019.8851920.

[33] *DeepSig RF Datasets for Machine Learning*. Accessed: Nov. 14, 2022. [Online]. Available: https://web.archive.org/web/20230304025639/https://www.deepsig.ai/datasets

[34] S. Mohanti, N. Soltani, K. Sankhe, D. Jaisinghani, M. Di Felice, and K. Chowdhury, "AirID: Injecting a custom RF fingerprint for enhanced UAV identification using deep learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[35] G. Reus-Muns, D. Jaisinghani, K. Sankhe, and K. R. Chowdhury, "Trust in 5G open RANs through machine learning: RF fingerprinting on the POWDER PAWR platform," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[36] N. Soltani, G. Reus-Muns, B. Salehi, D. Jennifer, S. Ioannidis, and K. Chowdhury, "RF fingerprinting unmanned aerial vehicles with non-standard transmitter waveforms," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15518–15531, Dec. 2020.

[37] *Radio Frequency Machine Learning Systems (RFMLS)*. Accessed: Mar. 22, 2023. [Online]. Available: https://www.darpa.mil/program/radio-frequency-machine-learning-systems

[38] *A Community Resource for Archiving Wireless Data at Dartmouth (CRAWDAD)*. Accessed: Mar. 12, 2023. [Online]. Available: https://crawdad.org/

[39] M. Rodrig, C. Reis, R. M. Mahajan, D. Wetherall, J. Zahorjan, and E. Lazowska, "CRAWDAD UW/SIGCOMM2004 (v. 2006-10-17)," IEEE Dataport, 2006. [Online]. Available: https://ieee-dataport.org/open-access/crawdad-uwsigcomm2004-v-2005-08-01, doi: 10.15783/C7SW2H.

[40] A. Al-shawabka, F. Restuccia, S. D'Oro, and T. Melodia, "Massive-scale I/Q datasets for WiFi radio fingerprinting," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107566, doi: 10.1016/j.comnet.2020.107566.

[41] M. S. Allahham, M. F. Al-Sa'd, A. Al-Ali, A. Mohamed, T. Khattab, and A. Erbad, "DroneRF dataset: A dataset of drones for RF-based detection, classification and identification," *Data Brief*, vol. 26, Oct. 2019, Art. no. 104313, doi: 10.1016/j.dib.2019.104313.

[42] (2021). *Machine Learning for Communications Emerging Technologies Initiative*. [Online]. Available: https://mlc.committees.comsoc.org/datasets/

[43] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1048–1077, 2nd Quart., 2021, doi: 10.1109/COMST.2021.3064259.

[44] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 2177–2184, doi: 10.1109/ICDCS.2017.283.

[45] A. K. Hagelskjær, B. H. Grevenkop-Castenskiold, M. H. Jespersen, T. Arildsen, E. Carvalho, and P. Popovski, "IoT device identification dataset," Zenodo, 2020. [Online]. Available: https://zenodo.org/record/3638165/export/hx, doi: 10.5281/zenodo.3638165.

[46] D. Anderson, *USB System Architecture*. Reading, MA, USA: Addison-Wesley Professional, 1997.

[47] L. J. Wong, S. McPherson, and A. J. Michaels, "An analysis of RF transfer learning behavior using synthetic data," 2022, *arXiv:2210.01158*.

[48] L. J. Wong, S. McPherson, and A. J. Michaels, "Assessing the value of transfer learning metrics for RF domain adaptation," 2022, *arXiv:2206.08329*.

[49] G. S. Gadgets. (2023). *YARD Stick One*. Accessed: Apr. 28, 2023. [Online]. Available: https://web.archive.org/web/20230418020802/https://greatscottgadgets.com/yardstickone/

[50] (2013). *Low-Power SoC (System-on-Chip) With MCU, Memory, Sub-1 GHz RF Transceiver, and USB Controller*. Texas Instruments. [Online]. Available: https://web.archive.org/web/20230503175809/https://www.ti.com/lit/ds/symlink/cc1110-cc1111.pdf

[51] Great Scott Gadgets. (2023). *YARD Stick One*. Accessed: Apr. 28, 2023. [Online]. Available: https://github.com/greatscottgadgets/yardstick

[52] *Numato Lab 8 Channel USB Relay Module*. Accessed: Feb. 3, 2023. [Online]. Available: https://web.archive.org/web/20230503180241/https://numato.com/product/8-channel-usb-relay-module/

[53] (2023). *USB Atmospheric Pressure, Temperature and Relative Humidity Sensor PTH200*. Dracal Technologies. Accessed: Apr. 3, 2023. [Online]. Available: https://web.archive.org/web/20230503175153/https://www.dracal.com/wp-content/uploads/2021/07/Dracal-PTH200-Datasheet.pdf

[54] (2022). *RfCat*. Accessed: Apr. 28, 2023. [Online]. Available: https://github.com/atlas0fd00m/rfcat

BRAEDEN P. MULLER (Member, IEEE) received the B.S. degree in computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2021, where he is currently pursuing the M.S. degree in computer engineering.

In 2021, he participated with the Virginia Microelectronics Consortium (VMEC) Summer Scholar Program performing research with the University of Virginia, Charlottesville, VA, USA. In 2022 and 2023, he was a Product Engineering Intern and a Digital Design Intern of Analog Signal Chain with Texas Instruments. He is currently a Graduate Research Assistant with the Virginia Tech National Security Institute, Blacksburg. His research interests include design and automated test of complex hardware systems.

LAUREN J. WONG received the B.A. degree in computer science and mathematics from the Oberlin College, Oberlin, OH, USA, in 2016, and the M.S. degree in electrical engineering from Virginia Tech, Blacksburg, VA, USA, in 2018, where she is currently pursuing the Ph.D. degree in electrical engineering.

From 2018 to 2020, she was a Research Associate with the Hume Center for National Security and Technology, Virginia Tech. Since 2020, she has been a Research Scientist with the AI Laboratory, Intel Corporation. Her research interests include various aspects of RFML, from applications to operational considerations and user assurance, and data efficient learning and AI robustness.

WILLIAM H. CLARK IV (Member, IEEE) received the M.S. and Ph.D. degrees in electrical and computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2015 and 2022, respectively.

He is currently a Research Scientist with the Spectrum Dominance Division, National Security Institute, Virginia Tech, where he works to improve various aspects of the RFML data space by improving the tools an intuition available to the domain.

ALAN J. MICHAELS (Senior Member, IEEE) received the degree in electrical and computer engineering, in applied mathematics, and in operations research from Georgia Tech, Atlanta, GA, USA, and the M.B.A. degree from Carnegie Mellon, Pittsburgh, PA, USA.

Previously, he was a Systems Engineer, a Researcher, and the Department Head of Harris Corporation, specializing in secure communication. He is currently a Professor with the Bradley Department of Electrical and Computer Engineering (ECE), Virginia Tech, and the Director of the Spectrum Dominance Division, National Security Institute, Blacksburg, VA, USA.

Dr. Michaels is a Professional Engineer with the Commonwealth of Virginia and has 44 issued U.S. patents. He is a fellow of the National Academy of Inventors.