Sitara: Spectrum Measurement Goes Mobile Through Crowd-sourcing

Phillip Smith[1,*], Anh Luong[2], Shamik Sarkar[1], Harsimran Singh[1],
Neal Patwari[1,3], Sneha Kasera[1], Kurt Derr[4], Samuel Ramirez[4]
[1]University of Utah, [2]Carnegie Mellon University, [3]Washington University in St. Louis, [4]Idaho National Labs
[*]Corresponding Author: phillip.smith@utah.edu

***Abstract*—Software-defined radios (SDRs) are often used in the experimental evaluation of next-generation wireless technologies. While crowd-sourced spectrum monitoring is an important component of future spectrum-agile technologies, there is no clear way to test it in the real world, i.e., with hundreds of users each carrying an SDR while uploading data to a cloud-based controller. Current fully functional SDRs are bulky, with components connected via wires, and last at most hours on a single battery charge. To address the needs of such experiments, we design and develop a compact, portable, untethered, and inexpensive SDR we call *Sitara*. Our SDR interfaces with a mobile device over Bluetooth 5 and can function standalone or as a client to a central command and control server. The Sitara offers true portability: it operates up to one week on battery power, requires no external wired connections and occupies a footprint smaller than a credit card. It transmits and receives common waveforms, uploads IQ samples or processed receiver data through a mobile device to a server for remote processing and performs spectrum sensing functions. Multiple Sitaras form a distributed system capable of conducting experiments in wireless networking and communication in addition to RF monitoring and sensing activities. In this paper, we describe our design, evaluate our solution, present experimental results from multi-sensor deployments and discuss the value of this system in future experimentation.**

***Index Terms*—Mobile Systems, Spectrum Monitoring, Wireless Networks, Crowd-sourcing, Software-Defined Radio**

I. Introduction

Future mobile wireless advancements will continue a trend of increasing densification, distribution and coordination, and spectrum-agile operation [14], [15], [17]. The performance of these new technologies depends not only on the mobility of individual users with respect to base stations, but also users' mobility with respect to each other. Ideally, to quantify performance experimentally, one would run a large-scale distributed wireless experiment with tens or hundreds of software-defined radios (SDR) programmed to deploy/test a new technology, while individual volunteers each carry these SDRs with them during their normal daily activities. Such an experiment would allow technologies to be tested with users' real-world mobilities, including temporal, spatial, and person-to-person correlations, rather than in artificial testbed or simulation environments that implicitly or explicitly assume independence and stationarity.

For researchers to be able to run such experiments, the SDR must be *truly portable*, so that a volunteer participant is not burdened by the carrying of the device, and in fact does not ordinarily notice it. Furthermore, the hardware must be *low cost* to enable measurements with hundreds of volunteers. We define portable to mean capable of operating for extended periods of time without an external power supply, small enough to carry without encumbering the user — small enough to easily fit into a pocket — and *not tethered* to wires, cables, or external connectors. Its energy consumption must allow it to last as long as most smartphones so that volunteers can charge it on the same schedule as their mobile device. In terms of cost, a researcher should be able to purchase a set of 100 on a standard grant, which would translate to around US $50 or less per device.

Unfortunately, no existing SDR meets the requirements of performing such experiments. Recent products include devices such as the Kickstarter-funded "portable SDR" (PSDR), RTL-SDR, HackRF, LimeSDR and Ettus USRP radios. While each of these possess useful capabilities, all fall short in one area or another for large scale mobile experiments. The most suitable among these would be the Ettus USRP E312, a battery-powered portable SDR [7]. The E312, however, is far too large to fit into a pocket and it must be tethered; it would still require a cable connection to a mobile device or laptop to provide a control channel. Additionally, the steep price of the E312 would present a practical limit to large scale distributed experiments. The RTL-SDR is low cost, but requires an external processor, to which it must be connected to by USB cable and, most importantly, has no transmit capability. Most true SDRs, capable of digital processing of RF samples and RX/TX, cost at least US $100 in the most basic form and quickly reach 10 times that cost for more sophisticated offerings. Such SDRs quickly become cost-prohibitive for large-scale experiments, and present a significant limitation shared by others [2].

Current mobile phones, while packed with cellular, Wi-fi and Bluetooth radios, do not always provide the flexibility to make arbitrary changes across layers which researchers want to explore, despite their suitability for some specific applications in localization and sensing [3], [18], [19]. Differences in chipsets, firmware implementation, protocols and carrier-imposed restrictions preclude uniform or arbitrary access and control of the underlying hardware. Even with unrestricted access to the hardware, such operation could disrupt data services and inconvenience the volunteer — something we wish to avoid.

A. A Novel Software-Defined Radio

Our contribution is a novel open-source device and cloud framework aimed at enabling large-scale experimental research in mobile dynamic spectrum access, propagation modeling, distributed and coordinated reception, and localization. Our device, called *Sitara*, is a truly portable software-defined radio. It is especially suited for distributed, crowd-sourced experiments. It is designed to have a battery life of up to a week on a single charge, to be smaller than a credit card, and to cost less than existing fully-featured SDRs. Our Sitara is convenient for volunteers to carry and is accessible to a broad set of researchers. We anticipate this to be particularly useful for scenarios in which simultaneous, near real-time, geographically distributed narrow-band RF measurements are desired. We will demonstrate how the Sitara can become a valuable tool, quickly amassing measurement inputs for models and providing insights to inform decisions for wireless research.

B. Achieving True Portability

The aim of achieving a compact, cordless, energy efficient device constrains key design decisions. The inconvenience of frequent charging and limited space for batteries make the power requirements of field-programmable gate arrays (FPGAs) commonly used in other SDR solutions unfeasible. For a device to be practical for crowd-sourcing it must also be convenient for volunteers to carry, which means we cannot connect it via cable to their smartphone, and little to no interaction should be required from the volunteer. With the recent availability of Bluetooth 5 devices and the ubiquity of smartphones, we arrive at the solution presented here: a low power transceiver paired with a Bluetooth interface. By pairing with the volunteer's phone, we can piggyback on the phone's WiFi or cellular connection to communicate with a remote server, as well as its location service.

But how can we avoid the large cost and size of most fully-functional SDRs? We apply a lesson from the RTL-SDR, which re-purposed a mass-produced digital video receiver (the RTL2832U) for its ability to output complex-baseband (IQ) samples. We use the Texas Instruments (TI) CC1200 transceiver which, although not designed as an SDR transceiver, has an IQ sample feature as well as transmit capability. This transceiver supports operation below 1 GHz. The experiments we perform in this paper are in the 902-928 MHz ISM band. Our firmware limits transmission to one of several ISM bands below 1 GHz. The Sitara complements the CC1200 with a Nordic Semiconductor nRF52840 system on chip (SoC) and supporting circuitry. This SoC contains a a processor and Bluetooth stack used for processing commands and communicating with a mobile device/gateway. The Sitara supports reading from and writing to arbitrary registers on the CC1200 radio, tuning radio frequency, measuring RSSI, continuously capturing IQ samples, sample capture on carrier-sense, frequency phase lock, and transmission and reception of messages using various modulations.

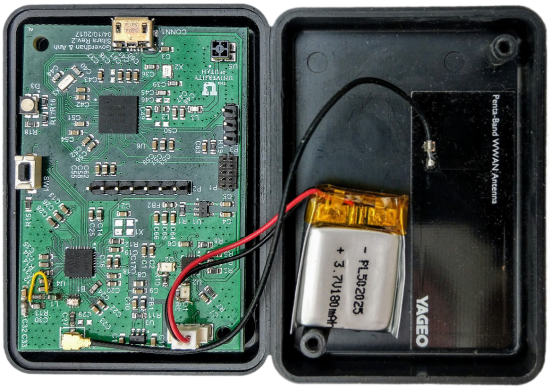

Fig. 1: Sitara PCB with antenna and rechargable battery housed in an ABS plastic enclosure

C. Balancing Power and Throughput

For any mobile device, the power budget is always an area of concern. For our application, while targeting low-cost, lower power components, we introduce the challenge of maintaining high sample throughput on hardware that was originally designed for intermittent, bursty operation; continuously operating the μC alone would deplete our initially specified battery in a matter of hours. We address this problem by designing an architecture that maximizes efficiency by exploiting hardware peripherals to maintain a high data rate while minimizing μC activity. To prove useful as an SDR, we must maintain a uniform sampling rate for IQ data. This requires solving a number of problems to achieve a careful coordination between transceiver data acquisition, sample processing and the Bluetooth radio.

In general terms, we achieve this by minimizing processing overhead in software and optimizing parameters for Bluetooth transmission. Among the Bluetooth features that make this possible are Low-Energy Data Packet Length Extension introduced in Bluetooth 4.2 and the optional 2 Mb/s bit rate, LE 2M PHY, introduced in Bluetooth 5 [1]. Because the Bluetooth stack is implemented as a "SoftDevice", a precompiled binary image, which runs on the single ARM core, there is inherent contention for the μC's resources. Any timing anomalies occurring while servicing interrupts by the SoftDevice result in a critical fault. Consequently, the SoftDevice must be given interrupt priority, resulting in non-deterministic timing for servicing other interrupts such as sample capture. Our solution overcomes these challenges to provide continuous sample capture over SPI and only requires μC intervention to rotate between receive buffers. The result of our efforts is a maximum, hardware-limited, continuous sampling rate up to 104 kS/s across the SPI interface and Bluetooth data rates exceeding 1Mb/s. The maximum sampling rate across the SPI interface effectively limits our receiver bandwidth to 52 kHz for IQ sampling.

D. Cloud-based Command and Control Server

In addition to the Sitara, we develop a mobile application and command and control server interface allowing hundreds of devices to operate in coordination, as shown in Fig. 2. The server provides a convenient web-based GUI for live monitoring and control of connected clients (Fig. 4) and processing of historical measurement data. This allows monitoring real-time measurements in a distributed, mobile environment or delayed logging and upload for later analysis. Accessible records contain RSSI measurements, IQ samples, location, time and device ID. These capabilities enable passive crowd-sourced measurements using remote control, or to function as a standalone SDR, controlled wirelessly through a user's mobile device. This is similar to prior efforts [2], but focuses on custom tailoring of experiments for a fine degree of control, rather than optimizing for one specific application.

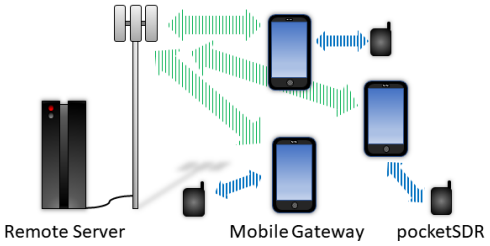

Fig. 2: Sitara backhaul system includes the Bluetooth connection (blue) between the Sitara and mobile gateway, and the WiFi or cellular connection (green) between the mobile device and the remote cloud server.

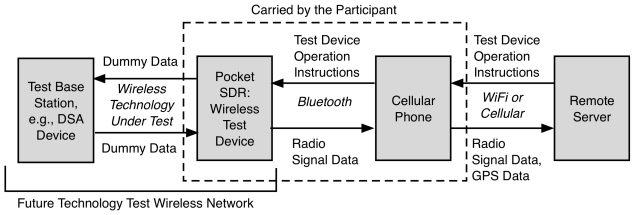

Fig. 3: Sitara measures signals from other Sitaras or other test devices, and uses the phone and its data connection and GPS to log data and receive commands.

In this paper we first cover the design and implementation, walking through solutions to some of the challenges we encountered. We characterize the performance of the Sitara then present experimental results from the following usage scenarios to illustrate the utility of our SDR:

- Transmitter localization using RSSI measurements
- Crowd-sourced measurements using multiple concurrent participants, suitable for spectrum monitoring or RF propagation modeling
- Server-side demodulation from IQ sample captures of a 2-FSK transmission

We conclude by assessing our solution and discussing future areas of research related to our crowd-sourced measurement approach.

II. Design and Implementation

In this section, we examine some of the technical challenges and design decisions during the development of the Sitara, beginning at the hardware component level and then continuing with the firmware development. We also briefly discuss the software development associated with the mobile gateway and server applications.

A. Component Choice

In order to develop a low cost, low power solution we look at transceivers capable of RF digital sample output. After considering many options we tended toward wireless transceivers such as the TI CC1200, Atmel RF-233, AT86RF215, Atmel AT86RF215IQ, and Silicon Labs EFR32FG. Among these, interface options and operational frequencies lead us to the CC1200 which tunes to frequency bands between 137 MHz and 950 MHz. The CC1200 is energy efficient and allows raw IQ samples to be exported while still operating over a wide enough frequency range to prove useful. The RF network can be configured to match the frequency and bandwidth of operation. In this paper, Sitara uses a 915 MHz balun, which places the optimal operating frequency between 902 to 928 MHz. Future development could add an RF switch or wide-band matching network to improve performance across other bands.

The CC1200 transceiver operates using a SPI interface to read and write data and control registers, respectively. General-purpose IO (GPIO) connections between the CC1200 and SoC μC allow interrupt-driven functions such as IQ sample acquisition and RF power level triggering. We adapt the register configurations for optimal spectrum monitoring. The CC1200 provides 3 registers (17 bits total) of magnitude and 2 registers (10 bits total) of angle measurements from the output of its coordinate rotation digital computer (CORDIC) algorithm.

We choose the nRF52840 SoC because it was one of the first available low-power Bluetooth 5 SoCs with a well-supported SDK. Additionally, the ARM Cortex-M4 within the SoC provides a floating point unit (FPU) which is necessary for some SDR applications. The RF output from the nRF52840 is connected to a 2.4 GHz 3dBi SMD chip antenna. While chip antennas are inefficient, the use case is to have a very short Bluetooth link, for example a volunteer might carry both devices in the same handbag, or in two different pockets. Such short links can be reliable even with the antenna loss as we can see from the Bluetooth throughput measurements in section III-A. A power management IC (PMIC) regulates voltage, charges and manages the LiPo battery connected through the standard JST connector. Sitara contains a JLink interface to allow programming, terminal logging and debugging. The CC1200's RF chain interfaces with an RF-tuned circuit terminating on a μFL connector. For our experiments and the results presented here, we use a Yageo Penta-Band WWAN antenna, but other antennas could also be used. The board, battery, and antenna are designed to fit within a standard 70 by 50 by 20 mm plastic case, which provides mechanical

protection while being carried by a volunteer. The battery is recharged by the volunteer using a standard micro USB cable, likely to be familiar to an Android phone user.

At the time of writing, the total cost for the bill of materials (BOM) in quantities of 1000 was estimated to be $38.00 per device. Please refer to our github repository to view the current BOM, source code and design documents [16]. Additional information about our system implementation can be found in arXiv:1905.13172 [cs.NI].

B. Sitara Firmware

We develop the SoC firmware using the nRF5 SDK v13.0.0 from Nordic Semiconductor, compiled using the GNU ARM toolchain v7.2.1. The firmware executable code resides in flash on-board the nRF52840 SoC. The SoftDevice, a pre-compiled protocol stack, is also stored in flash and loaded into RAM at run-time. An event-driven API allows the firmware to interface with the SoftDevice to access Bluetooth functions.

Once powered on, the μC initializes and configures the external CC1200 transceiver, on-chip Bluetooth radio and other peripherals then enters a sleep state while awaiting commands. As we mention, minimizing power consumption is a key design driver, so minimizing the time that components are powered on and active is a recurring theme. This allows us to achieve an 80% power reduction for most applications.

Most commands perform a single function then return the μC to a sleep state. The continuous SAMPLE CAPTURE command enters a loop in which data is acquired and sent via the Bluetooth interface. Because continuous sample capture is an important aspect of our design we will discuss its operation in more detail.

Sample capture utilizes the Programmable Peripheral Interconnect (PPI), which permits on-chip peripherals to interact through task-event relationships, independent of the CPU. We configure an interrupt event associated with the magnitude-valid output signal from the CC1200 to trigger a burst-read SPI transaction task which reads the registers containing the sample data. The magnitude-valid signal asserts when a new IQ sample is ready on the CC1200 and occurs at a set rate dependent on the configured receiver filter bandwidth. While most of these actions are automated using hardware peripherals, the nRF52840 SoC, unfortunately, does not provide a method of switching receive buffers for the SPI-DMA transaction without μC intervention; however, the time required to service the interrupt is approximately 22 μs. Therefore, sampling rates with a period equal to or greater than 22 μs (45 kS/s) may be delayed by at most 1 sample period. An additional factor that complicates this process and limits the maximum sampling rate over SPI is the absence of a hardware-enabled control function which allows consecutive burst reads from multiple registers (although this is available for repeated reads from a single register). This negatively impacts performance in two ways:

- Each sample read must include two command bytes for SPI burst read which are added to the total length of the transaction, thus increasing each transfer duration.
- During a SPI transfer, a byte is received during each clock cycle, thus the receive buffer will always contain the two status bytes received during the clock cycles which the two command bytes are sent, in addition to the bytes containing the actual data.

This results in inefficient use of memory and a receive buffer containing status bytes interleaved with sample data requiring extra processing steps to extract samples. Nevertheless, this does not impact performance because the Bluetooth connection ultimately limits throughput as we note in Section III-A.

Bluetooth transfer of packets is not real-time, and due to packet collisions and errors, MAC delays and retransmissions, any finite buffer can experience an overflow. This is handled by pausing capture acquisition and discarding samples while the Bluetooth interface catches up. This is problematic for RF sample capture because it can break continuity and introduce timing offsets. In order to compensate for these errors, we maintain a 16 MHz counter during acquisition which is activated while sample capture is paused and then reports the elapsed pause time once sample capture resumes. This information is then also sent over Bluetooth so the end-user is able to accurately reconstruct and preserve timing of sample captures.

C. Crowdsourcer and Server Software

The choices of server architecture and software frameworks were driven primarily by convenience, ease of use and adaptability rather than resource optimization as we see for the Sitara. The server application uses common tools and frameworks including Python Flask and an Apache front-end paired with Gunicorn, a Python WSGI server. The control functions are accessed from an interactive Javascript-based GUI in a web browser. Socket.IO libraries for Python, Javascript and Java provide low-latency, standardized event-based communication between different the server, Web GUI and the device gateways, respectively.

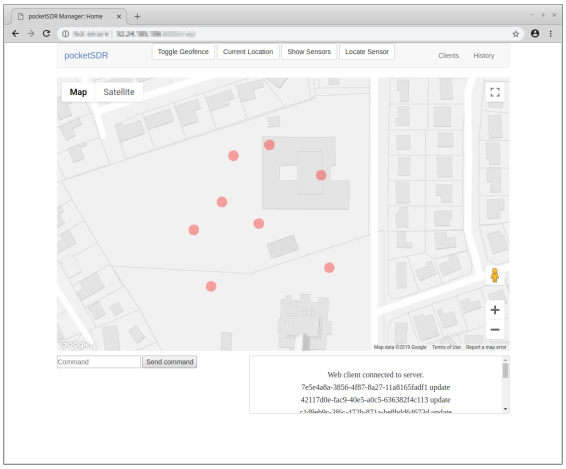

Fig. 4: Server homepage showing locations of active sensors

A remote operator issues commands from a web client which are processed by the server and relayed to the ap-

propriate client device gateways. The web client, server, and gateway each activate a set of event listeners which filter relevant messages. Messages containing measurement results in response to commands, such as RSSI and IQ data, are stored in the server's database. The web interface provides a Google Maps overlay which can display real time client location and associated measurement data as shown in Fig 4. In addition to real-time monitoring of sensors, the web client also provides convenient tools for filtering and displaying subsets of measurements according to parameters such as time, frequency, RSSI threshold and location. Some of these capabilities will be demonstrated in section IV.

The Server application was designed to provide a high-level abstraction for commands, leaving the implementation details and low-level commands to the client device gateways. For example, a server RSSI command emits a single message containing several parameters such as frequency, bandwidth, reporting interval and report type. The device gateway receives this message and issues multiple commands as appropriate, such as frequency tune and RSSI capture, over Bluetooth to individual Sitara devices. A Sitara in turn will interpret each of these commands and perform the appropriate functions to accomplish this task. Combining and abstracting commands at the top level in this way reduces latency, preserves bandwidth and spares server resources by minimizing the number of message exchanges for a given task. This abstraction also provides a modular approach making the sensor implementation-agnostic to the server.

While developing this architecture, we considered implementing an existing standard such as IEEE 802.22.3 Spectrum Characterization and Occupancy Sensing (SCOS) Sensor [13], an extension of the SigMF specification [8]. We find that such specifications provide a level a complexity beyond our immediate needs. Specifically, SCOS implements a restful API requiring each sensor to host a web server; we instead use an open Socket.IO protocol to maintain persistent connections necessary for managing mobile devices. Notwithstanding the differences, our implementation could be adapted to comply with the SCOS standard as follows: Rather than implementing a RESTful API on each sensor, a gateway could be installed on the server which is compatible with the SCOS standard and emulates the API for each device. As requests are received at this server, the SCOS gateway would forward the appropriate commands to the server application. This would provide all the benefits of compatibility with the SCOS standard without a system redesign.

III. Evaluating Our Solution

We now present measurements characterizing the performance of our system under varying conditions. A small sample size used to obtain reasonable expectation of performance.

A. Data Throughput

Three data paths potentially limit the real-time throughput of the Sitara system. The Bluetooth data rate, the SPI interface from the CC1200 radio to the μC, and the on-board μC itself. Depending on the use case, any of these could become a bottleneck. In our application, the maximum data transfer across the SPI bus exceeds the data rate of the Bluetooth link.

The CC1200 specifications limit the minimum SPI clock rate to 7.7 MHz for extended register reads, which include the registers of interest. In practice we have successfully achieved an 8 MHz clock rate. Two common modes of operation read from the CC1200 either three magnitude and two angle registers (8-bits each) or only the two angle registers. The maximum achievable sampling rates for these two modes were 64 kS/s and 104 kS/s, respectively.

The Bluetooth 5 standard defines a maximum transfer rate of 2 Mbps [1]. By utilizing this LE 2M PHY option, packet length extension and configuring the Bluetooth Maximum Transmission Unit (MTU) to match the packet length, we achieved highest throughput. Initial throughput testing using special firmware achieved a peak effective data rate of approximately 1.3 Mbps between a Sitara device and Bluetooth 5 capable mobile phone. In typical usage scenarios we observe an average throughput greater than 1 Mbps. This determination results from a series of tests under varying environmental conditions using two different mobile phones, denoted *device 1* and *device 2* in Fig. 5. For each of these measurements, 244KB of data was transmitted over the Bluetooth link while the Sitara recorded the transmit time on its system clock. The 244KB transmission was then repeated at least 10 times for each test. For the environmental test (Fig. 5, top) we used two controls, the first places a mobile phone within 15 cm of the Sitara in an environment without in-band WiFi or Bluetooth activity as observed on a spectrum analyzer. The second control repeats this test, but separates the paired devices by 12 meters. We then perform additional measurements as follows: with the participant keeping the Sitara in a pocket while holding the phone in hand (Test 1), with the Sitara inside a backpack while the participant holds the phone in hand (Test 2), and finally, in a variety of different indoor and outdoor environments with the Sitara in the participant's pocket (Test 3). In addition to the environmental tests, we also conduct interference tests (Fig. 5, bottom). For these experiments, up to four interfering devices transmit over Bluetooth at their maximum data rate while the unit under test performs measurements. For these tests, the paired devices are separated by one meter. The controls consist of the same test, with no interferers (Control 1) and four interferers (Control 2); though in the case of the latter, the separation is reduced to 15 cm while other conditions remain the same. Test 3 of the interference measurements notably exhibits higher deviation than others. We speculate that this is due to characteristics of the Bluetooth protocol at marginal capacity or differences in hardware among devices. Further analysis of the Bluetooth performance and characterization is beyond this scope of this work as these tests are intended only to approximate worst-case conditions expected in real-world scenarios.

In our application, minimal processing was required by the onboard μC during sample acquisition and was not found to impact throughput, therefore no attempt was made to evaluate

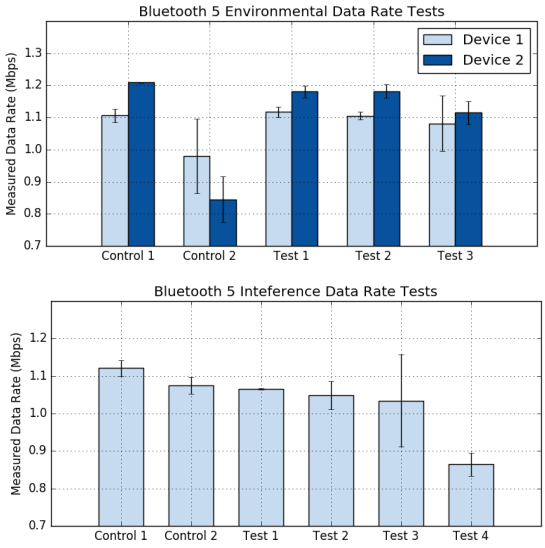

Fig. 5: Measured Bluetooth 5 data rate between Sitara and two mobile devices under varying test scenarios (top). Measured Bluetooth 5 data rate in the presence of varying numbers of interferers, where test number refers to the number of interferers (bottom). All error bars represent standard deviations.

load or processor utilization on the Sitara. If additional signal processing were to be carried out on the Sitara, then this may require further investigation.

B. Power Characteristics

The Sitara's power consumption ranges from approximately 18 mW to 180 mW depending on operational mode. Power usage in typical scenarios is shown in Fig. 6 below. In the Sitara idle state, the CC1200 radio is set to a low powered state, maintaining power only to the crystal oscillator and digital core; when no other commands are present the μC in the nrRF52840 chip receives the wait-for-event (WFE) command which likewise powers down nonessential modules. The μC will periodically wake up to handle events necessary to maintain a Bluetooth connection. During RX and TX states, the CC1200 remains powered on along with necessary RF, clock and interface peripherals; the μC is more heavily utilized in these states, but no quantitative analysis was performed to determine the duty cycle of the active versus inactive/WFE state.

We initially tested battery life with a 3.7 V 180 mAh LIPO battery, allowing the Sitara to operate for several hours in any state. Repeated tests demonstrate that at full charge, the Sitara maintains a consistent Bluetooth connection until loss of power after 37 hours; this is consistent with the estimated battery life based on the measured idle power consumption. We later opted for a larger 850 mAh battery to extend battery life up to a week with a commensurate improvement in high duty-cycle operation. For comparison, the USRP E312 SDR uses a 3200 mAh battery to achieve 5.5 hours at idle [6].

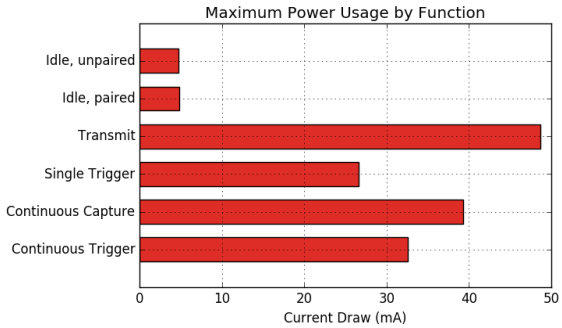

Fig. 6: Comparison of maximum Sitara energy consumption across a range of operational states.

C. Timing and Synchronization

The CC1200 contains a phase-locked loop (PLL) which allows timing recovery in hardware for available modulation schemes. In other applications, our firmware performs a phase-lock function to effectively tune to a signal of interest. Regular synchronization may be necessary to maintain clock accuracy in use cases such as Doppler, time-of-arrival, or synchronous RF network architectures.

For system timing and timestamp information relating to signal measurement, the Sitara maintains a system clock. The Sitara can synchronize its clock by receiving a clock command containing current GPS time from the mobile gateway over Bluetooth. Multiple such clock commands are sent until the mobile gateway observes a minimum round-trip time between the sent time and command acknowledgement — a technique not very different from many network time protocols. This provides a simple method of synchronizing devices with GPS time with an error on the order of milliseconds. In order to achieve a more accurate synchronization, Sitara devices can perform a triggered RSSI measurement which will upload a timestamp associated with the trigger event to the server. From a collection of triggered RSSI measurements on the server for a single transmit event, relative clock offsets between different devices can then be easily computed. If an application requires a more accurate node synchronization, then a more sophisticated approach would be appropriate [9], [11].

D. Server Performance

Although some performance compromises are necessary to meet our hardware design goals for the Sitara, server-side resources present few practical constraints as they are easily re-configurable at run-time. By hosting our server on a cloud-computing service, our application can quickly scale to meet demand. As a base configuration we reserve a host with 1 vCPU and 0.5 GB of RAM. We find this adequate to display and manage 250 simultaneous clients each reporting one RSSI measurement per second. Additional clients reporting RSSI, or multiple clients uploading IQ samples may require a more capable server if the live reporting mode is desired. As our system is expected to handle dropouts and latency associated

with mobile data links, we place no hard timing requirements on server resources and assume best-effort.

IV. Case Studies

In this section, we present experimental results demonstrating how our system is used in two different applications. We specifically choose these applications to demonstrate the versatility and key features of our system namely: the portability of the Sitara for crowd-sourced measurement and its utility as an SDR. We also explain how these demonstrations can extend to much more complicated experiments.

Note that for experiments involving volunteers, to ensure consent and handling of potentially sensitive user data is adequately addressed, we maintain an institutionally-approved IRB.

A. Spectrum Sensing

One of our primary design goals for the Sitara is to offer a convenient platform for crowd-sourced spectrum sensing. As such, we demonstrate the versatility of our system in this capacity in real-world settings.

1) Transmitter Localization: Single Sitara: In this scenario, we deploy one Sitara acting as a receiver which is controlled locally using an open-source third-party mobile application: *nRF UART v2.0*. The user walks around the test site, issues commands from the mobile gateway to capture RSSI measurements which are later used to estimate the location of a "rogue" transmitter.

Multiple RSSI measurements are taken to estimate the location of a receiver. The measurement points are chosen arbitrarily along pedestrian-accessible paths. The measurement points used for localization are indicated by green circles overlaid onto the site map in Fig. 7 (Left). The radius of the green circles are proportional to the measured RSSI value at each of these points. The red circle denotes the transmitter location. The test site covers a roughly 28,000 m^2 area, with the farthest discernible measurement captured at a distance approximately 120 m from the transmitter.

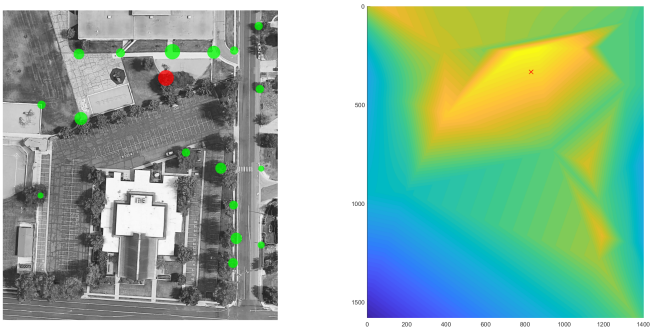

Fig. 7: (left) Outdoor test area with transmitter location (●) and measurement locations (●), with radius proportional to the measured RSSI value. (right) Transmitter location estimation map and true location (×).

Once the data is captured, the coordinates of the measurement points are associated with individual pixels of a captured overhead image of the test site retrieved from Google Maps. To make localization more challenging, we choose to discard the highest three RSSI measurements. The remaining points are used to produce Fig. 7 (left), then as inputs to a Matlab interpolant object from which Fig. 7 (right) is generated. In this case, linear interpolation is used. We can see that this small data set performs reasonably well in locating the transmitter. The true location is approximately 9 m away from the best estimate as indicated by the peak of the yellow region in the figure. Clearly with more measurements, either from an increased sampling rate or multiple simultaneous devices, we would see this accuracy improve.

Here, we deploy a single Sitara with a local, active participant to capture RSSI measurements at different points. Local Sitara operation may be convenient for directed experiments which may not accommodate multiple passive users in a crowd-sourced scenario. A more typical use case will involve multiple participants–active or passive–and rely on automated, server-initiated measurement commands. We demonstrate these capabilities next.

2) Crowd-sourcing: Multiple Sitaras: We deploy twelve Sitaras among passive participants in a series of experiments. By passive we mean the participants are not directed but walk around freely while the nodes are operated remotely from the server, requiring no interaction from participants. By automating different test scenarios using server-side scripts, we are able to rapidly acquire large volumes of data. Fig. 8 depicts path loss in a suburban environment calculated from RSSI measurements and GPS coordinates of multiple devices obtained using a round-robin transmit scheme, in which individual nodes take turns operating as a transmitter while others measure RSSI. This is a fast and efficient method to generate data for inputs into complex propagation models that may otherwise be difficult to obtain [12]. Fig. 9 (left) presents a server-generated overlay of RSSI measurements obtained from twelve Sitaras scanning a range of frequencies over a user-specified duration. Here, each point radius scales relative to RSSI and each color, again, represents a unique device.

Fig. 9 (right) shows a server-generated overlay from prior measurements stored in the database using an automated command script. This image is generated by querying the server for measurements (blue dots) from a specific device between two time points. These measurements are not intended to locate a known transmitter, but instead demonstrate the passive crowd-sourcing capability. Regardless, if our object was to determine locations of possible sensors, we could request another overlay including multiple devices and an RSSI threshold. Along one path in the figure, we also see a drop in measurements, this likely indicates a loss of mobile data service in that particular area.

We present these examples to demonstrate the utility of the Sitara for conducting distributed spectrum measurement experiments. In Section V, we discuss ways we can adapt our platform to accommodate more sophisticated experiments.

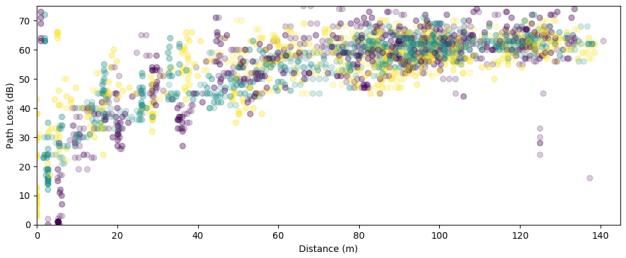

Fig. 8: A snapshot in time showing path loss using RSSI measurements from multiple Sitaras, distinguished by color, as a function of distance from a transmitter. Using Sitara, such data sets are easily generated and can be used for propagation models.

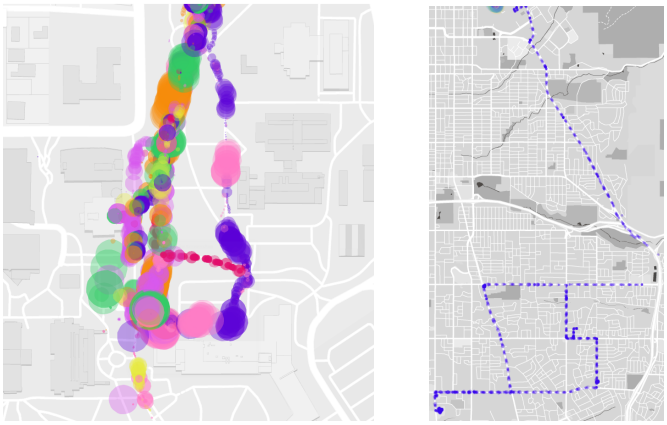

Fig. 9: Server-generated overlays of remote RSSI measurements from twelve devices carried by participants over a series of experiments, where each color represents a unique device and each point radius is scaled relative to RSSI value (left) and server generated overlay of RSSI measurements from one device over a wide-area (right).

B. Server-side Processing with GNU Radio

In this application, we report the capability of the Sitara to capture IQ samples from an over-the-air FSK transmission, upload the raw data to a remote server and recover the original message by demodulating the signal in software using GNU Radio. While this may superficially appear to be an unnecessarily complex method of performing a relatively simple demodulation, we present this as a proof of concept and later discuss the compelling implications when extended to future areas of research.

The experimental setup uses multiple Sitaras, one acting as the transmitter, while the others receive. The user gives the transmitter a transmit command which directs the Sitara's radio to send a short message consisting of a preamble, sync word and user-defined payload, using the CC1200's native 2-FSK modulation format. The receiving Sitaras, instead of demodulating the signal directly in hardware, are configured to perform a triggered capture command while a carrier signal is detected. Once the sample capture is complete, the IQ samples, consisting of 16 bit integers, are uploaded to the server where further processing occurs. In this case, we use a simple GNU Radio flow-graph to demodulate the signal (Fig. 10) and extract the payload message from one of the captures. A plot of the angle samples captured from different receivers can be seen in Fig. 11.

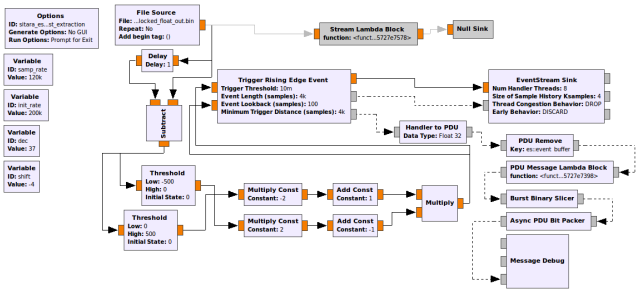

Fig. 10: A GNU Radio flow-graph used to generate the 2-FSK demodulator script

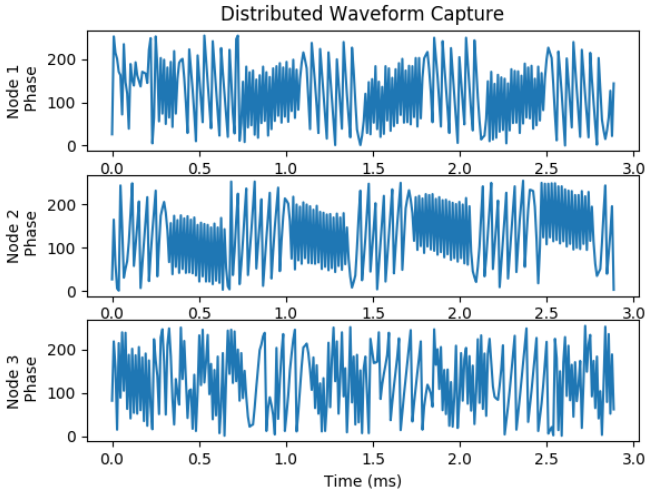

Fig. 11: Waveform phase measurements obtained from a triggered waveform capture using three Sitaras

Although only one captured signal was needed to demodulate the incoming signal in this case, using multiple Sitaras, several copies of the same waveform captured at different geographical locations can be uploaded to the server for more sophisticated signal processing. In Fig. 11 the waveforms do not appear to be synchronized, but the relative timing information can be recovered using the capture timestamp and correlating between waveforms. This example demonstrates the Sitara's capability to capture an RF signal and upload the data to the server for cloud-based reception. We want to enable experimentation in coordinated mobile multi-antenna reception, using centralized cloud computing, to be able to separate signal from interference and demodulate signals that may not be recoverable from any one receiver alone. Such

experiments have been demonstrated to be powerful with static receivers [5], and Sitara can enable such experiments with crowd-sourced mobile endpoints.

V. Discussion

A. Follow-on Efforts

We plan to expand our framework to test more sophisticated localization algorithms and develop propagation models enabled by our rapid data gathering capability across many sensors. Hundreds of participants can accumulate millions of pairwise propagation measurements between different transmitting and receiving nodes to generate new, highly accurate trained models. Extending the server-side GNU Radio application, we can apply Sitara to experiments in distributed, coordinated signal processing as proposed by others [4], [5], [9]. Our system can be adapted to operate as a mesh network with other devices or a testbed for additional network architectures. By altering the network topology and allowing a single mobile gateway to act as a central device to multiple Sitaras, and improving clock synchronization, we may leverage diversity gain to perform experiments in distributed multiple input multiple output (MIMO) and coherent combining at each node. Beyond these technical applications, in a future work we plan on investigating incentives, privacy and other participant-related issues which are beyond the scope of this paper.

B. When should Sitara not be used?

Having presented a number of applicable use cases for our system, there remain areas where the Sitara is not particularly well suited. The CC1200 transceiver and RF front end limit the frequencies in which Sitara can operate. Additionally, the maximum sampling rate and Bluetooth throughput constrain the SDR to narrowband operation. This is a direct consequence of the design choices made to minimize cost and maximize portability via a wireless back-channel. A more sophisticated receiver would require an FPGA or another ASIC with a much higher clock frequency and a delicate analog RF front-end. This would not only substantially reduce battery life, but also increase device cost. Our platform is designed to operate in ISM bands where transmission is permissible and operating frequency is inherently restricted; it is not intended to be a wideband receiver. Despite these limitations, there still exist many other applications where Sitara would be a suitable test platform [10].

VI. Conclusion

The absence of viable options for large scale, coordinated, crowd-sourced spectrum sensing motivated our development of Sitara, which we present here. We characterize the system and highlight its advantages for distributed spectrum measurement activities. We promote our design based on its merits: (1) Energy efficiency, with a battery life lasting up to one week — sufficient for a broad range of experiments. (2) An inexpensive, compact form-factor including a wireless backhaul, offering an ideal solution for mobile, crowd-sourced scenarios. (3) Capability of local, manual or automated, and remote operation of sensors within a network distributed across a wide geographical area. (4) SDR capabilities to measure complex temporal and spatial RF interactions. We showcase the Sitara's capabilities in real-world scenarios and evaluate its performance. The Sitara is a valuable open-source resource for research in distributed software-defined radio sensing.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants No. 1564287 and No.

References

[1] Bluetooth SIG. *Bluetooth Core Specification*, December 2016. v5.

[2] A. Chakraborty, M. S. Rahman, H. Gupta, and S. R. Das. Specsense: Crowdsensing for efficient querying of spectrum occupancy. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[3] E. C. L. Chan. *Introduction to wireless localization : with iPhone SDK examples*. 2012.

[4] B. V. den Bergh, D. Giustiniano, H. Cordobés, M. Fuchs, R. C.-P. S. Pollin, S. Rajendran, and V. Lenders. Electrosense: Crowsourcing spectrum monitoring. *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN 2017)*, pages 1–2, March 2017.

[5] A. Dongare, R. Narayanan, A. Gadre, A. Luong, A. Balanuta, S. Kumar, B. Iannucci, and A. Rowe. Charm: Exploiting geographical diversity through coherent combining in low-power wide-area networks. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '18, pages 60–71, Piscataway, NJ, USA, 2018. IEEE Press.

[6] M. Ettus. Universal Software Radio Peripheral. [Online].

[7] Ettus Research. *Getting Started with the Ettus Research USRP E312 SDR*, March 2016. Rev 1.

[8] GNU Radio Foundation. *Signal Metadata Format Specification*, 7 2018.

[9] E. Hamed, H. Rahul, and B. Partov. Chorus: Truly distributed distributed-mimo. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 461–475, New York, NY, USA, 2018. ACM.

[10] G. Hattab and D. Cabric. Spectrum sharing protocols based on ultra-narrowband communications for unlicensed massive iot. In *2018 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 1–10, Oct 2018.

[11] J. He, P. Cheng, L. Shi, and J. Chen. Time synchronization for random mobile sensor networks. volume 63, pages 2712–2717, 12 2012.

[12] Y. S. Meng, Y. H. Lee, and B. C. Ng. Empirical near ground path loss modeling in a forest at vhf and uhf bands. *IEEE Transactions on Antennas and Propagation*, 57(5):1461–1468, May 2009.

[13] NTIA. *IEEE 802.22.3 Spectrum Characterization and Occupancy Sensing*, 1 2018. v0.0.2.

[14] M. Shafi, A. F. Molisch, P. J. Smith, T. Haustein, P. Zhu, P. D. Silva, F. Tufvesson, A. Benjebbour, and G. Wunder. 5g: A tutorial overview of standards, trials, challenges, deployment, and practice. *IEEE Journal on Selected Areas in Communications*, 35(6):1201–1221, June 2017.

[15] S. K. Sharma, T. E. Bogale, L. B. Le, S. Chatzinotas, X. Wang, and B. Ottersten. Dynamic spectrum sharing in 5G wireless networks with full-duplex technology: Recent advances and research challenges. *IEEE Communications Surveys Tutorials*, 20(1):674–707, Firstquarter 2018.

[16] SPAN lab. Sitara. https://github.com/SPAN-UofU, 2019.

[17] C. Wang, F. Haider, X. Gao, X. You, Y. Yang, D. Yuan, H. M. Aggoune, H. Haas, S. Fletcher, and E. Hepsaydir. Cellular architecture and key technologies for 5G wireless communication networks. *IEEE Communications Magazine*, 52(2):122–130, February 2014.

[18] W. Xue, w. qiu, X. Hua, and K. Yu. Improved wi-fi rssi measurement for indoor localization. *IEEE Sensors Journal*, PP:1–1, 01 2017.

[19] M. Zhou, Z. Tian, K. Xu, X. Yu, and H. Wu. Theoretical entropy assessment of fingerprint-based wi-fi localization accuracy. *Expert Systems with Applications*, 40(15):6136 – 6149, 2013.