A Software-Defined Ultrasonic Networking Framework for Wearable Devices

G. Enrico Santagati, *Student Member, IEEE*, and Tommaso Melodia, *Senior Member, IEEE*

***Abstract*— Wearable medical devices with wireless capabilities have become the cornerstone of many revolutionary digital health applications that promise to predict and treat major diseases by acquiring and processing physiological information. Existing wireless wearable devices are connected through radio frequency electromagnetic wave carriers based on standards, such as Bluetooth or Wi-Fi. However, these solutions tend to almost blindly scale down traditional wireless technologies to the body environment, with little or no attention to the peculiar characteristics of the human body and the severe privacy and security requirements of patients. We contend that this is not the only possible approach, and we introduce U-Wear, the first networking framework for wearable medical devices based on ultrasonic communications. U-Wear encloses a set of physical, data link, and network layer functionalities that can flexibly adapt to application and system requirements to efficiently distribute information between ultrasonic wearable devices. U-Wear also offers reconfiguration functionalities at the application layer to provide a flexible platform to develop medical applications. We design two prototypes that implement U-Wear and operate in the near-ultrasonic frequency range using commercial-off-the-shelf (COTS) speakers and microphones. Despite the limited bandwidth, i.e., about 2 kHz, and COTS hardware components not optimized for operating at high frequency, our prototypes: 1) achieve data rates up to 2.76 $kbit/s$ with bit-error-rate lower than 10^{-5} using a transmission power of 13 dBm (20 mW); 2) enable multiple nodes to share the medium; and 3) implement reconfigurable processing to extract medical parameters from sensors with high accuracy.**

***Index Terms*— Body area networks, body sensor networks, wearable medical devices, ultrasonic communications and networking, acoustic communications and networking.**

I. Introduction

WEARABLE medical sensing and actuating devices with wireless capabilities have become the cornerstone of many revolutionary digital health applications [2]. Wearable electrocardiography (ECG) devices and blood pressure sensors can enable remote cardiovascular monitoring for early diagnosis of cardiac arrhythmias and hypertension [3], and therefore prevent heart failures and strokes. Skin patches with wireless connectivity can be arranged in a closed-feedback-loop drug delivery system [4]. For instance, a sensor patch can measure the level of subcutaneous blood glucose, while a drug pump patch can adaptively deliver the required amount of insulin. Motion sensors, e.g., accelerometers, can collect large amounts of data for fitness and medical applications. For example, wireless motion trackers can record step rate, speed, and acceleration for performance monitoring. Similar devices can also collect information for post-surgery telerehabilitation in case of lower-limb injuries or strokes [5], or measure the motor degradation of patients affected by *Parkinson* disease [6]. Likewise, by correlating motion with sleep activity, the same sensors can monitor the REM sleep duration and provide information on the development of *post-traumatic stress disorders* [7].

Existing wireless wearable medical devices are connected through radio frequency (RF) electromagnetic waves. Standards in use are often scaled down versions of wireless technologies (e.g., Bluetooth and Wi-Fi), with little or no attention to the peculiar characteristics of the human body and to the severe privacy and security requirements of patients. For example, many commercial activity trackers, as well as smart watches, and smart clothing [8], [9] connect to smartphones using Bluetooth or Wi-Fi. Alternatively, other medical monitoring solutions [10] use proprietary RF-based technologies to collect medical data in a centralized manner. We contend that this is not the only possible approach, and that RF-based technologies have several limitations that can negatively affect the patient experience with wearable devices.

Limitations of RF Technology: The RF frequency spectrum is scarce and already crowded with many devices interfering with one another. At the same time, the number of wireless devices that compete to access the RF spectrum is growing exponentially. This includes wireless sensors, but also more largely and pervasively deployed RF devices, and even microwave ovens. A large number of medical devices operate in the unlicensed industrial, scientific, and medical (ISM) band [11] because it is freely available, internationally regulated and largely adopted. Healthcare facilities generally use the ISM band for medical telemetry, nurse-call systems as well as patient data links. Moreover, public and private Wi-Fi networks and personal Wi-Fi hotspots are pervasively deployed in hospitals, substantially crowding the ISM spectrum. Quoting the FDA's recent guideline on wireless medical devices, "an increasingly crowded RF environment could impact the performance of RF wireless medical devices" [12]. Therefore, RF-based technologies raise serious concerns about potential interference from existing RF communication systems that can unintentionally undermine the reliability and security of the wearable network, and ultimately the safety of the patient. Moreover, RF communications can be easily

Manuscript received December 18, 2015; revised May 20, 2016 and July 18, 2016; accepted September 12, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Liu. Date of publication October 28, 2016; date of current version April 14, 2017. This work supported by the National Science Foundation CAREER under Grant CNS-1253309.

The authors are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: santagati@ece.neu.edu; melodia@ece.neu.edu).

Digital Object Identifier 10.1109/TNET.2016.2616724

jammed, i.e., intentionally disrupted by artificially generated interference, or eavesdropped by malicious agents using cheap and off-the-shelf equipment.

Based on these observations, in this paper, we propose to use ultrasonic waves to interconnect wearable devices, and present U-Wear, the first software-defined framework for creating networks of wearable medical devices based on ultrasonic communications. Ultrasonic waves are acoustic waves with frequency higher than the upper threshold for human hearing (nominally $20\ kHz$). Acoustic waves have found application in underwater communications [13], in airborne applications such as indoor localization [14] and gesture recognition systems [15], and, massively, in ultrasonic medical imaging [16]. Finally, ultrasonic waves have been proven to be a viable alternative to RF waves for creating wireless networks of implantable medical devices [17]–[19].

Benefits of U-Wear: U-Wear has several benefits over traditional networking based on RF communications:

i) U-Wear eliminates any potential conflict with existing RF systems and overcrowded RF environments.
ii) The ultrasonic frequency spectrum is unregulated and enables nodes to flexibly adapt the occupied frequency to specific requirements such as maximum level of tolerable co-channel interference, maximum tolerable channel multipath and Doppler spreading in the channel and minimum data rate needed at the application layer, among others.
iii) Compared to RF waves, ultrasonic waves do not easily penetrate through solid materials and do not propagate far in air; therefore, ultrasonic communication systems are inherently more secure with respect to eavesdropping and jamming attacks that would require close proximity between the attacker and the victim.
iv) U-Wear is completely implemented in software and runs on general-purpose hardware; thus, it allows developers to implement U-Wear on commercial devices, such as smartphones, laptops and smart-TVs, and enable these devices to communicate in the near-ultrasonic frequency range, i.e., $17 - 22\ kHz$, using commercial-off-the-shelf (COTS) speakers and microphones [20], without requiring external hardware components.
v) The U-Wear software-defined functionalities can be reconfigured to adapt to application requirements, offering more flexibility with respect to RF-based networking systems entirely implemented in hardware.

U-Wear consists of a set of software-defined cross-layer functionalities designed to network ultrasonic wearable devices that offer real-time reconfigurability at different layers of the protocol stack, i.e., the physical (PHY), data link, network and application layer, and can be implemented and run on general-purpose hardware such as microprocessors, microcontrollers, and FPGAs. Specifically, U-Wear encloses a set of PHY, data link and network functionalities defined in software that can flexibly adapt to the application and system requirements to efficiently distribute information among wearable devices. U-Wear also offers real-time reconfigurability at the application layer to provide a flexible platform to develop medical applications. In particular, sensor data processing applications running in the nodes are decomposed into *primitive building blocks* that can be arbitrarily arranged to create new sensing applications that fit user requirements, and can be installed on the device and run on the fly without requiring firmware updates.

As a proof of concept, we design two prototypes that implement the U-Wear framework and operate in the near-ultrasonic frequency range, i.e., $17 - 22\ kHz$, using COTS speakers and microphones. Specifically, we operate at $18\ kHz$ with a bandwidth of about $2\ kHz$. Despite the use of COTS audio components not optimized for operations at high frequency, our prototypes (i) can achieve data rates up to $2.76\ kbit/s$ with bit-error-rate (BER) lower than 10^{-5} at a transmission power of $13\ dBm$ ($20\ mW$); (ii) enable multiple nodes to share the same medium, with tradeoffs between packet delivery delay and packet drop rate; and (iii) implement reconfigurable data processing to extract medical parameters from sensors with high accuracy. Moreover, U-Wear proposes for the first time the use of a Gaussian minimum-shift keying (GMSK) signaling scheme for the near-ultrasonic frequency range that allows to achieve relatively high data rates when compared to previously proposed near-ultrasonic systems; more importantly, it ensures virtually inaudible[1] click-free transmission because of the GMSK phase-continuity.

Note that we do not claim improved performance as compared to RF in terms of traditional performance metrics (e.g., data rate, bit error rate, energy consumption). Any such comparison would clearly be premature - current performance of RF systems is the result of many years of research and development in multiple fields and of a multi-billion dollar industry. Yet, we believe that, for the reasons discussed above, ultrasonic networking is a viable alternative, and in this paper we present a proof of concept and explore several system tradeoffs. The remainder of this paper is organized as follows. In Section II, we discuss the fundamentals of ultrasonic communications along the body. In Section III, we introduce U-Wear, the first networking framework for wearable medical devices, and discuss its architecture. In Sections IV and V, we present the design and implementation of two prototypes, and we thoroughly discuss the performance of U-Wear through experimental results. In Section VI, we discuss related work, and finally, in Section VII, we conclude the paper.

II. Ultrasonic Communications in Air

Ultrasounds are mechanical pressure waves that propagate through elastic media at frequencies above the upper limit for human hearing, i.e., $20\ kHz$.

Attenuation: Two mechanisms mainly contribute to acoustic attenuation in air: i.e., spreading loss and absorption loss [21]. The former includes spherical spreading, i.e., the acoustic pressure falls off proportionally to the surface area of a sphere. The latter is mainly related to atmospheric absorption caused by the interaction of the acoustic wave with the gas molecules of the atmosphere, and is frequency-, temperature-, and humidity-dependent.

[1]The upper threshold for human hearing is nominally $20\ kHz$, however hearing starts degrading significantly above $15\ kHz$.

For a signal at frequency $f\,[Hz]$ over a transmission distance $d\,[m]$, the attenuation can be expressed in $[dB]$ as

$$A_{dB} = 20\log_{10}(d) + d\,\alpha(f), \tag{1}$$

where $\alpha(f)$ $[dB/m]$ is the absorption coefficient, which increases quadratically with the frequency, but also depends on the atmospheric pressure, temperature, and humidity [22].

Propagation Speed: The propagation speed of acoustic waves in air is approximately 343 m/s at $20°C$ and at atmospheric pressure of 101.325 kPa, as compared to $3\times10^8\,m/s$ for RF waves. The speed of sound in air increases with temperature and humidity, going from 331 m/s at a temperature of $0°C$ and 10% relative humidity to 351 m/s at a temperature of $30°C$ and 90% relative humidity.

Doppler Spreading: Doppler spreading occurs as a result of Doppler shifts caused by relative motion between source and receiver, and is proportional to their relative velocity. Doppler spreading generates two different effects on signals: a frequency translation, and a continuous spreading of frequencies that generates intersymbol interference (ISI), thus causing degradation in the communication performance. Since the speed of sound is several orders of magnitude lower than the speed of electromagnetic waves, the resulting Doppler effect is severe, even at relatively low speeds.

Reflections and Scattering: The on-body ultrasonic channel is composed of several interfaces between air and human body, and between air and on-body and near-body objects. Because of this inhomogeneous pattern, the on-body channel can be modeled as an environment with pervasive presence of *reflectors* and *scatterers*. Consequently, the received signal is obtained as the sum of numerous attenuated, possibly distorted, and delayed versions of the transmitted signal.

Air-Coupled Ultrasonic Transducers: An ultrasonic transducer is a device that converts electrical signals into ultrasonic signals and vice versa. A piezoelectric transducer produces a mechanical vibration through a thin piezoelectric element under an external voltage variation, and produces a voltage variation under an external mechanical vibration.

When the operating frequency of the ultrasonic communications falls in the near-ultrasonic frequency range, i.e., $17-22\ kHz$, acoustic waves can be recorded and generated using COTS components, such as microphones and speakers. Even though COTS components are often designed to operate at lower frequencies, i.e., $0-17\ kHz$, they can still sense and generate, albeit less efficiently, near-ultrasonic frequency waves. Since many commercial devices such as smartphones, tablets and laptops among others, are equipped with audio interfaces, they can support near-ultrasonic communications with no additional hardware [20].

III. U-Wear Architecture

U-Wear consists of a set of software-defined multi-layer functionalities that can be implemented on general-purpose processing units, e.g., microprocessors, microcontrollers or FPGAs, among others, to enable networked operations between wearable devices equipped with ultrasonic connectivity, i.e., air-coupled ultrasonic transducers, and sensing capabilities, i.e., sensors. Figure 1 shows an overview of the U-Wear framework. U-Wear runs on a processing unit. It accesses an analog-to-digital converter (ADC) and digital-to-analog converter (DAC) through hardware-specific system APIs. In the transmit (Tx) chain, the DAC collects and digital-to-analog converts U-Wear's outputs, i.e., the waveforms to be transmitted, before passing these to the communication unit. In the receive (Rx) chain, the ADC analog-to-digital converts and passes to U-Wear the received waveforms coming from the communication unit. The communication unit consists of an ultrasonic transducer and an amplification stage, i.e., preamplifier in Rx chain and power amplifier in Tx chain. U-Wear also collects the analog-to-digital converted data coming from the sensing unit.

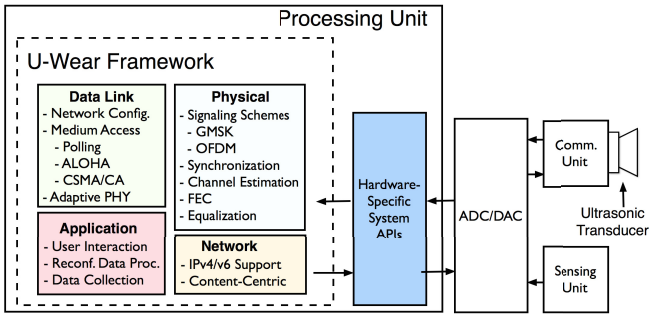

Fig. 1. Overview of the U-Wear framework.

A. Physical Layer

U-Wear PHY layer libraries define the signaling scheme, channel estimation, equalization, synchronization and forward error correction (FEC) functionalities.

1) Signaling Schemes: U-Wear offers two fully-functional signaling schemes, a narrowband scheme based on GMSK modulation, and a wideband scheme based on orthogonal frequency-division multiplexing (OFDM). U-Wear also includes a set of software-defined primitive blocks, e.g., filters, and Fast Fourier Transform (FFT) modules, that can be used to implement additional signaling schemes.

Narrowband GMSK: GMSK is a continuous-phase modulation (CPM) used worldwide in GSM cellular systems [23]. In frequency-shift keying (FSK) and phase-shift keying (PSK), information is encoded in the variations of the carrier frequency, or carrier phase, respectively. Since frequency and phase switches occur instantaneously, FSK and PSK signals do not have continuous phase. Phase discontinuity generates out-of-band power, leading to poor spectral efficiency. Moreover, in near-ultrasonic transmissions based on COTS speakers and microphones the out-of-band power introduces audible noise (clicks), which makes the communication perceptible to humans. Differently, GMSK signals have phase continuity, and enables click-free transmissions. The product between the signal bandwidth B and the symbol time T is a measure of the spectral efficiency of the scheme. Lower BT product leads to higher spectral efficiency, but increases the intersymbol interference (ISI). The BT product can be regulated as desired to increase the spectral efficiency by keeping the ISI under control. This feature makes GMSK the signaling scheme of

choice for narrowband communications in the near-ultrasonic frequency range based on COTS speakers and microphones.

Wideband OFDM: OFDM has been extensively used in underwater acoustic communications [24] because of its robustness against frequency-selective channels with long delay spreads. The idea behind OFDM is to use a large number of closely spaced orthogonal sub-carriers, such that for each sub-carrier the channel is subject to flat fading. In each sub-carrier a conventional modulation is used, e.g., M-PSK or M-Quadrature-Amplitude-Modulation (QAM). OFDM offers high spectral efficiency and robustness against narrowband co-channel interference, intersymbol interference (ISI) and multipath fading. Finally, OFDM can be efficiently implemented using FFT and inverse FFT (IFFT) algorithms. These characteristics make OFDM ideal for ultrasonic communications based on wideband transducers.

2) Synchronization: Synchronization is achieved by correlating the received signal with a local copy of the preamble, i.e., a sequence that precedes each packet, which outputs a peak corresponding to the first sample of the packet. U-Wear offers two synchronization modes, which are chosen to address two main propagation challenges of ultrasonic in-air communications, multipath and Doppler effect.

PN-Sequence Mode: The pseudo noise (PN)-sequence mode uses PN-sequences as a preamble, i.e., binary sequences with sharp autocorrelation peak and low cross-correlation peaks, which can be deterministically generated. Specifically, we consider maximum length sequences (MLSs), a family of PN-sequences that can be generated in software and hardware through linear feedback shift registers (LFSRs). Because of their desirable correlation characteristics, PN-sequences have been widely used to enable strong resilience to multipath [25]. Therefore, they are well suited for ultrasonic in-air communications, as discussed in Section II.

Chirp-Based Mode: The chirp-based mode uses a chirp signal as preamble, i.e., a sinusoidal waveform whose frequency varies from an initial frequency f_0 to a final frequency f_1 within a certain time T. Chirp signals have been widely used in radars [26] due to their good autocorrelation and robustness against Doppler effect. In fact, a frequency-shifted chirp still correlates well with the original chirp, although with lower amplitude and time-shifted peak. This characteristic makes chirp synchronization desirable in ultrasonic in-air communications under severe Doppler effect conditions as experienced, for example, under fast movement conditions as in sensors worn by athletes. The price we pay for the Doppler robustness is higher cross-correlation peaks compared to PN-sequences that result in lower resilience to multipath.

3) Channel Estimation, Equalization and Forward Error Correction: U-Wear implements channel estimation, equalization and forward error correction (FEC) functionalities to estimate the channel impulse response (CIR), mitigate the distortion produced by the channel and correct potential transmission errors caused by the multipath and Doppler spread of the ultrasonic in-air channel, Section II.

U-Wear offers a training-based channel estimation approach that leverages the good autocorrelation property of the synchronization preamble sequence, discussed in Section III-A.2, to estimate the CIR. By correlating the output of the channel, i.e., the received signal, with the input, i.e., the known preamble sequence, we obtain an estimate of the time-domain CIR [27]. U-Wear also implements a linear equalization technique, zero-forcing (ZF) [28], that aims to minimize the ISI signal distortion produced by the channel. As the name suggests, a ZF equalizer is a finite-impulse-response (FIR) filter of order N that, for each input symbol, “forces to zero" the ISI introduced by the $2N$ adjacent symbols. Finally, U-Wear offers forward error correction (FEC) functionality based on Reed-Solomon (RS) codes, i.e., linear block error-correcting codes widely used in data storage and data transmission systems. An RS encoder takes k information symbols and adds t *parity* symbols to make an n symbol block. The RS coding rate is defined as k/n. An RS decoder decode the received n-symbol block, and can correct up to $\frac{t}{2}$ data symbols that may contain potential errors due to the channel fluctuation or collisions with interfering packets.

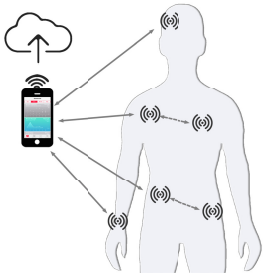

Fig. 2. U-Wear scenario with both M/S (continuous line) and P2P (dashed line) network configurations.

B. Data Link Layer

The U-Wear data link layer provides a set of functionalities that allow multiple nodes to access the medium under the challenges posed by the ultrasonic in-air channel, e.g., long propagation delays, among others, as discussed in Section II.

1) Network Configuration: U-Wear is designed to inter-network wearable devices in master/slave (M/S), peer-to-peer (P2P) configurations, or hybrid M/S and P2P configurations. Figure 2 shows a hybrid configuration system design. For example, *M/S configuration* may be used in continuous monitoring systems where a master node, e.g., a smartphone or a laptop, is used to fetch, analyze and display data collected by wearable sensors. Wireless or wired Internet connectivity may allow the master node to connect the wearable network with a medical center where the patient’s data can be stored, and analyzed remotely. *P2P configuration* suits, among others, applications that require distributed coordination among nodes for closed-feedback-loop operations. For example, this may include a skin patch drug-delivery system where a drug pump can trigger a sensor for measurement, or where a sensor may trigger the drug pump for drug injection after a measurement.

2) Medium Access Control Protocols: U-Wear offers three fully-functional multiple access protocols: i) a deterministic access protocol, i.e., polling; ii) a random access protocol, i.e., ALOHA [29]; and iii) a carrier detection protocol,

i.e., carrier sense multiple access (CSMA) with collision avoidance (CA) [30]. U-Wear also offers a set of primitive functions to implement custom protocols, e.g., idle listening, random backoff, or checksum-based error control mechanisms.

3) PHY Layer Adaptation: U-Wear defines a set of cross-layer functionalities that enable real-time reconfiguration of PHY layer parameters from upper layers of the protocol stack, e.g., data link or network layer. By leveraging the flexibility of the software-defined architecture, upper layer protocols can reconfigure on-the-fly PHY layer parameters such as modulation, signal bandwidth and FEC coding rate, among others. Reconfiguration functionalities allow developing reactive or proactive control algorithms to adapt the underlying communication link to the channel variations or to upper layer protocol requirements [19].

C. Network Layer

U-Wear provides interoperability with the Internet by defining an adaptation layer that integrates *IPv4 and IPv6 protocol support* [31]. The adaptation layer consists of a set of functionalities that interface the traditional IP network layer with the U-Wear data link layer, by offering IP header compression and IP packet fragmentation functions optimized for ultrasonic wearable networks with long propagation delays discussed in Section II that potentially prevent accurate timing of network protocols. For example, by leveraging cross-layer header information, the long IPv4 and IPv6 headers can be shortened to reduce network delay and energy consumption when exchanging small information packets.

U-wear also offers *content-centric networking functionalities* that make the network data directly addressable and routable. Each sensor data or actuation command, i.e., each *content object*, is labeled with a name, and can be accessed through this name.

D. Application Layer

1) Reconfigurable and Modular Data Processing: U-Wear adopts the idea of decomposing the data processing applications running in the sensor nodes into *primitive blocks*, and offering real-time reconfigurability at the application layer. The sensing application consists of a sequence of basic operations that are executed on the sensed data to extract desired medical parameters. Real-time modular reconfiguration offers two main advantages. First, the network coordinator can wirelessly transmit and install new applications on sensor nodes at runtime, as needed. Based on this, resources are allocated only when the application is requested, thus reducing the processing and memory overhead due to static applications continuously running in background. Second, modular reconfiguration enables programmers to easily create new applications by arranging the primitive building blocks in the desired execution sequence. As a consequence, new medical parameters can be extracted from the raw data coming from a sensor, while maximizing code reusability. Defining new applications consists of specifying inputs, a chain of primitive blocks, and outputs. An input is the physical sensor that generates the data, e.g., electrocardiogram (ECG). An output can be either the local memory for storing a measured parameter, or a transmission for sending a measured parameter to another node. We divide the set of primitive blocks into three main classes: i) *filters* enable filtering the raw data to remove offsets, drift or noise from the input signal; ii) *data operations* include common signal processing operations performed on sensor data, e.g., correlation and FFT, among others; and iii) *detectors* allow measuring the desired parameters by detecting specific elements in the processed signal, e.g., peaks, patterns and time distances, among others.

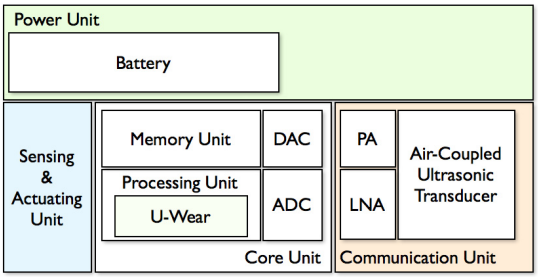

Fig. 3. Proposed hardware design of a wuMote.

IV. U-Wear Prototypes

The first U-Wear prototype is a wearable ultrasonic sensor node based on a custom hardware platform, which we refer to as *wuMote*. The second prototype is a wearable ultrasonic coordinator based on an iOS commercial smartphone device, which we refer to as *wuMaster*.

While commercial wireless system are commonly implemented in dedicated and optimized hardware components, e.g., application-specific integrated circuits (ASICs) or digital signal processors (DSPs), U-Wear protocols and transmission scheme functionalities are fully implemented in software, running on a resource constrained processing unit, i.e., a microcontroller and on the microprocessor of a commercial smartphone. Based on this, in the following, each protocol function is decomposed in software modules, which are designed and optimized to run with minimal resources, e.g., clock cycles and memory.

A. wuMote Prototype

1) Hardware Design: In Fig. 3, we show the hardware architecture of the wuMote. The *core unit* includes a processing unit, e.g., microprocessor or microcontroller, a memory unit, e.g., RAM or flash memory, and digital-to-analog and analog-to-digital converters. The *processing unit* executes the U-Wear functionalities discussed in Section III. The *communication unit* enables ultrasonic wireless connectivity by embedding power and low noise amplifiers, and air-coupled ultrasonic transducers. The *power unit* consists of a battery to power the wuMote. Finally, the *sensing and actuating unit* can incorporate several sensors and actuators according to the specific application design.

We implemented a prototype of the architecture in Fig. 3 based on Teensy 3.1 [32]. The wuMote implementation offers near-ultrasonic capability, by using COTS audio speakers and

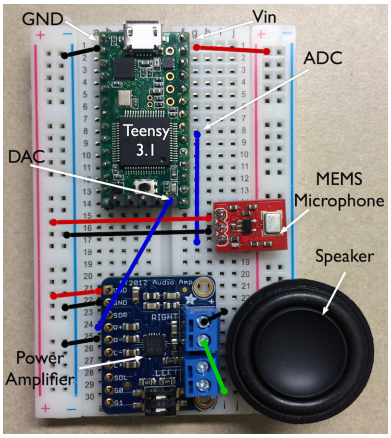

Fig. 4. Circuit diagram of the wuMote prototype.

microphones as air-coupled ultrasonic transducers. Figure 4 shows the basic circuit design of the wuMote prototype on a solderless breadboard. The prototype includes a Teensy 3.1, i.e., the core unit, a power amplifier, a microphone with low noise amplifier, and a small audio speaker, i.e., the communication unit. A lithium ion polymer battery, not shown in the figure, is connected to the bus strip of the breadboard to power the electronic components.

Teensy 3.1: Teensy 3.1 is a small-footprint, i.e., about 3.5×1.8 cm, inexpensive development board, based on a 32-bit ARM Cortex-M4. It comes with 64K of RAM, 256K of Flash, 12-bit DAC, dual ADC, and USB connectivity. The ARM Cortex-M4 can run at 24, 48 or 96 MHz. Teensy 3.1 can be programmed in C and C++ using Teensyduino, a customized version of the Arduino integrated development environment (IDE), and supports many of the code libraries designed for Arduino and others specifically designed for Teensy, e.g., the audio library, among others.

Ideally, lower-power and smaller-packaged solutions would be desirable for a more stable product. In a final product, the Cortex-M4 currently used in our prototype would likely be replaced by a Cortex-M0 microcontroller such as the mm-size low-power Freescale KL03 microcontroller. Even though development boards for Cortex M0 microcontrollers exist, none of these are comparable to the Teensy platform in terms of hardware and software capabilities, and they do not offer Arduino library support. Thus, at this stage we selected the less energy efficient Teensy 3.1 hardware platform in exchange for shorter prototyping time.

Power Amplifier: The wuMote includes a small and efficient class D audio amplifier able to deliver a maximum of 1 W into $4 - ohm$ impedance speakers, with a voltage supply of 3.3 V DC, and efficiency up to 80%. The amplifier consumes less than 2 mA of current when quiescent and less than 2 μA in standby mode. In Fig. 4, the right channel of the power amplifier is connected to Teensy via the DAC pin, and to the speakers via the 3.5 mm screw-terminal blocks. The Vcc and GND pins are connected to the bus strip to power the device.

Microphone: The wuMote includes a tiny breakout board that embeds an ADMP401 MEMS microphone and a low-noise amplifier (LNA). The ADMP401 offers a mostly flat bandwidth, i.e., -3 dB roll off, between 100 Hz and 15 kHz,

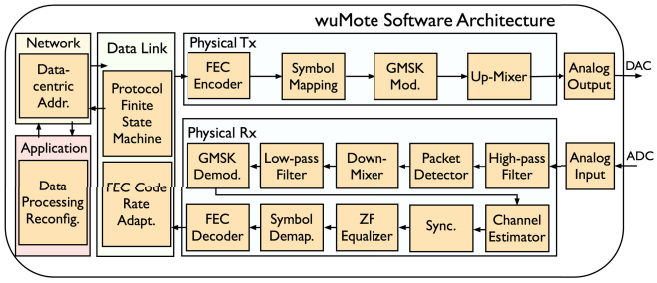

Fig. 5. Software architecture of the wuMote prototype.

omnidirectional sensitivity pattern, and requires a supply voltage between 1.5 V and 3.3 V DC. Although a microphone with larger bandwidth would perform better, we selected ADMP401 because of the COTS breakout board package that eases prototyping. Moreover, even though with lower sensitivity, the ADMP401 can still detect higher frequency acoustic waves up to 22 kHz. The microphone is connected to one of the analog pins (ADC) available in Teensy 3.1.

Audio Speaker: The output of the wuMote is a small and compact COTS speaker, Dayton Audio CE28A-4R, with $4 - ohm$ impedance, 4 W maximum output power supported, and flat frequency response between 100 Hz and 15 kHz. The speaker is connected to the power amplifier using 3.5 mm screw-terminal blocks.

2) Software Architecture: We implemented the U-Wear framework in Teensy 3.1 to enable ultrasonic wireless connectivity and networking on the wuMote hardware prototype. In Fig. 5, we show the block diagram of the wuMote software architecture that includes (i) narrowband GMSK transceiver with synchronization, channel estimation, equalization, and FEC functionalities at the PHY layer, (ii) polling and ALOHA multiple access protocol with FEC coding rate reactive adaptation at the data link layer, (iii) content-centric addressing at the network layer, and (iv) data processing reconfiguration with fetch and push support at the application layer.

We implemented the U-Wear functionalities using Teensyduino, an add-on for the Arduino IDE, leveraging many of the code libraries available for the Arduino platform. Since ultrasonic waves are nothing but sound at higher frequencies, we based the PHY layer signal processing on the audio library specifically designed for Teensy 3.1. The Teensy audio library consists of a set of objects that enable recording, processing, and playback of audio sampled at 44.1 kHz. Objects instantiate specific audio functionalities, e.g., a waveform synthesizer and finite-impulse-response (FIR) filters, and creating new object can enable new functionalities. A cascade of objects forms a processing chain that performs a set of operations on inputs to produce a desired output. Each object in the chain operates in pipeline on chunks of 128 audio samples, which correspond to 2.9 ms of audio. To guarantee audio continuity, each block must execute its processing operation within 2.9 ms, which can be a very stringent constraint for heavy computations such as filters or cross-correlations. In the wuMote implementation we built custom-made objects that implement specific signal processing operations.

Physical Tx: The first object of the PHY layer Tx chain is the `FEC Encoder`. Here, each data packet coming from

the data link layer is coded, as discussed in Section III-A.3, and overhead symbols are appended to the original packet. We select $n = 255$ symbols and parity symbols t to achieve different coding rates. Because of the computation complexity of RS coding, the `FEC Encoder` is implemented as an *off-the-chain* object. The coded packet is then passed to the `Symbol Mapping` object that inputs the audio stream in the processing chain. Here, the coded packet is serialized, i.e., converted into a stream of bits, differentially encoded, and transformed into a non-return-to-zero (NRZ) signal. The NRZ signal is then GMSK modulated by the `GMSK Modulator` object and up-converted to the carrier frequency by the `Up-Mixer` object. The modulated and up-converted waveforms are passed to the `Audio Output` object, i.e., a system API that interfaces U-Wear with the DAC, digital-to-analog converted and transmitted through the audio speaker.

Physical Rx: The received acoustic signal is converted into an electrical signal by the MEMS microphone. The signal is amplified by the LNA, and analog-to-digital converted by the Teensy 3.1 ADC at $44.1\ kHz$. The `Audio Input` object is a system API that interfaces U-Wear with the embedded Teensy 3.1 ADC, and inputs the audio stream into the PHY layer Rx chain. The received digital signal is first high-pass filtered by the `High-pass Filter` object to eliminate low-frequency noise and interference, i.e., external human voice and ambient noise. The `Packet Detector` processes the input signal to detect incoming packets using an energy-based approach to check whether there is energy at the expected carrier frequency. The incoming packet is then down-converted by the `Down-Mixer`, i.e., converted into a complex in-phase/quadrature baseband signal, and low-pass filtered to eliminate undesired higher-frequency harmonics introduced by the nonlinearity of the down-conversion. Channel estimation, synchronization and equalization operations normally follow down-conversion and are applied to the complex baseband signal. However, these operations are computationally expensive, and their execution exceeds the audio library time constraints of $2.9\ ms$. To overcome this limitation, we first demodulate the complex baseband signal in the `GMSK Demodulator` object to extract the phase variation that carries the coded information bits. The `Channel Estimator` object estimates the CIR using the packet preamble as training sequence, while the `Synchronizer` object attempts to achieve fine synchronization through the PN-based mode. The `ZF Equalizer` object filters the input for ISI recovery. The equalized symbols are demapped into a bitstream, collected into a packet structure, and passed to the `FEC Decoder` object, which attempts to correct potential bit errors. Finally, the error-corrected packet is passed to the data link layer.

Data Link Layer: The wuMote data link layer is implemented in a Finite State Machine (FSM) block that currently includes two of the MAC protocols discussed in Section III-B, i.e., polling and ALOHA. The wuMote data link layer also implements a PHY layer adaptation to optimally select the FEC coding rate that minimizes the number of retransmissions. During packet transmission, the MAC FSM collects data from upper layer protocols and creates the data-link-layer packet. The packet is then forwarded to the PHY layer Tx chain, where it is encoded in a digital waveform before being transmitted. At the receiver side, the MAC FSM detects the received packet based on information coming from the `Packet Detector` block and triggers the PHY layer to start processing the received waveform.

Network Layer: The wuMote network layer implements the content-centric addressing scheme discussed in Section III-C. Although IPv4 and IPv6 support is an important aspect of the U-Wear framework, at this prototyping stage we left most IP-related aspects out of the scope of this work. Each *content object* is mapped into a binary mask, where it is represented by the bit position in the mask. In an M/S configuration, a node joining the network is first paired with the master. The master maps the content available in the new node into the binary mask, and broadcasts the updated mapping to all the nodes in the network. Each node builds a local mask based on the entities that it possesses. To request an entity, the master broadcasts a request message containing a *request mask* with '1' set in the position mapped to the desired content.

Application Layer: The wuMote application layer implements the real-time modular reconfiguration functionalities discussed in Section III-D. Based on this modular approach, applications can be represented by chains of binary sequences, i.e., keys. Each primitive function is mapped to a binary key. A concatenation of keys represents a concatenation of operations, and therefore represents an application. The application is encapsulated into reconfiguration packets and transmitted over-the-air. The receiving node extracts the binary keys, and feeds these into an FSM where each state represents a primitive block function. By parsing the consecutive function keys, the FSM transitions from state to state, processing inputs and producing outputs. Inputs and outputs of each function are mapped to binary keys as well. The input and output keys allow to parametrically pass data between functions. As a proof of concept, we developed some applications based on the primitives discussed above.

Electrocardiogram Processing: We consider a single-electrode ECG signal. Fig. 6 shows a template signal with five labeled characteristic waveforms, P, Q, R, S and T, that correspond to electrical events during one heart beat. The first application measures the heart rate in beat-per-minute [bpm]. This is done following two different approaches. The *R method* counts the number of peaks, R *waves*, in a 6-second trace and multiplies the result by 10. The *RR method* finds the number of heartbeats per second by inverting the average of the *RR interval* duration, i.e., distance between two consecutive R waveforms in the trace, and multiplies this by 60. This approach has higher complexity with respect to the R-method, but results in higher resolution, i.e., 1 bpm against 10 bpm of the R method. The second application measures average and standard deviations of temporal separation between electrical events in the ECG. The mean and standard deviation of the RR interval, i.e., the distance between R peaks, provide information about potential heart arrhythmias [33]. Figure 6 (top) shows the simplified primitive block sequences of the R method heart rate detector and the RR interval monitor applications.

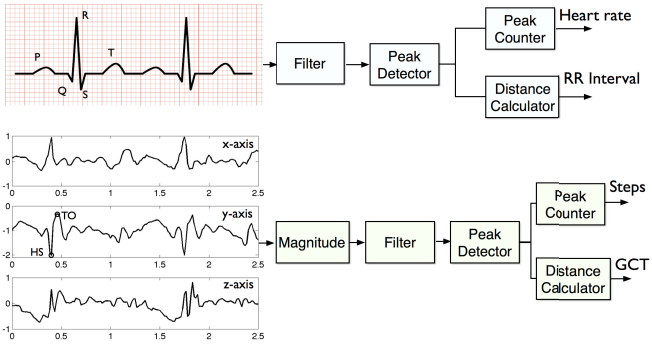

Fig. 6. Primitive blocks of heart rate and RR interval monitor.

Accelerometer Processing: Fig. 6 (bottom) shows a three-dimensional accelerometer trace. We label in the y-axis two main events that occur during a single step, i.e., *heel strike* (HS) and *toe-off* (TO), that correspond to the instants at which the foot touches the ground, and the instants at which the foot leaves the ground, respectively. Based on this, the first application calculates the magnitude of the acceleration from the three-axis components, low-pass filters it to remove high frequency noise, and counts the number of peaks in the resulting signal, i.e., the number of HSs. The peaks within a time interval represent the number of footsteps performed by the patient. The second application measures average distances between non-consecutive peaks in the acceleration magnitude, i.e., the time between consecutive HSs on the same foot (*gait cycle time*). Gait cycle time offers a measure of the motor degradation of patients affected by *Parkinson* disease [6].

B. wuMaster Prototype

We implemented the wuMaster prototype on the iOS 8 platform for Apple iPhone smartphones. The prototype consists of an app running on an iPhone that implements the U-Wear multi-layer functionalities. The software architecture of the wuMaster prototype is similar to the software architecture of the wuMote, and it includes (i) narrowband GMSK transceiver with synchronization, channel estimation, equalization, and FEC functionalities at the PHY layer, (ii) polling and ALOHA multiple access protocol with FEC coding rate reactive adaptation at the data link layer, (iii) content-centric networking at the network layer. At the application layer, the reconfigurable data processing is replaced with a graphical user interface (GUI) and speech recognition functionalities that allow users to interact with the wearable network.

The iOS prototype wirelessly communicates with the wuMotes through an ultrasonic narrowband GMSK link, using the phone's embedded microphone and speaker. We developed the software prototype in Objective-C programming language using Xcode 6 integrated development environment (IDE). As for the wuMote implementation, in iOS processing delay and memory constraints were also significant. Using a high-level programming language such Objective-C to implement heavy signal processing operations is not straightforward, and requires several design and optimization cycles to meet the desired system requirements. To simplify this process

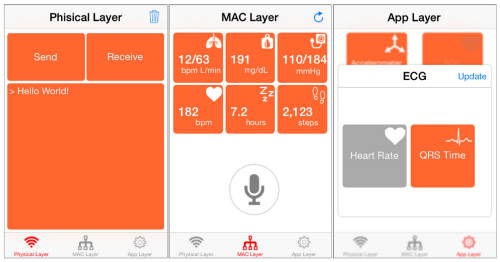

Fig. 7. wuMaster GUI for PHY, MAC and application layer.

we relied on three iOS libraries for audio and natural language processing. We use (i) the *vDSP library* [34] of the iOS Accelerate framework that provides several digital signal processing operations (DSP) to perform arithmetic operations and correlations on real and complex vectors in the PHY layer; (ii) *Novocaine* [35], a high performance audio library that enables record/play audio functionalities and hides all the low-level audio implementation details; and (iii) the *wit.ai* framework [36] that provides language processing, which we use to integrate voice command in U-Wear.

PHY Layer Tx/Rx: The PHY layer Tx and Rx are implemented in two classes named `PHYLayerTx` and `PHYLayerRx`, respectively. Here, the Novocaine Audio Manager triggers the `InputBlock` and `OutputBlock` callbacks to record and play audio, and the vDSP functions process the input and output data. At the transmitter, the `PHYLayerTx` class gets the data from the data link layer, generates the GMSK waveform, and then passes it to the Audio Manager. The latter transmits the GMSK waveform through the speaker. At the receiver, the operations in `PHYLayerRx` match those implemented in the wuMote PHY layer, discussed in Section IV-A.2. Because of the less stringent memory and processing constraints of the iOS platform, here channel estimation, synchronization and equalization follow the down-conversion, and are applied to the complex baseband signal.

Data Link and Network Layer: The wuMaster data link layer implements polling and ALOHA MAC protocols, as well as FEC coding rate adaptation. The MAC functionalities are implemented in a class named `MACLayer`, where a FSM implements the MAC protocol operations. The wuMaster network layer implements the same content-centric addressing scheme seen in the wuMote prototype, with the exception that here the centralized mapping functionalities are also implemented.

Application Layer: The wuMaster application layer consists of a GUI and a set of wit.ai commands that allow users to interact with the U-Wear multi-layer functionalities. The GUI's principal element is a three-tab `TabViewController` class, in Fig. 7. The first tab (left) contains a `PHYViewController` object used to test the PHY layer performance. The second tab (center) contains a `MACViewController` object that allows the user to test the MAC Layer functionalities by requesting, and receiving data coming from the deployed wuMotes, e.g., number of footsteps, sleep hours, and heart rate, among others The third tab (right) contains an `APPViewController` object that gives access to application layer reconfiguration functionalities discussed

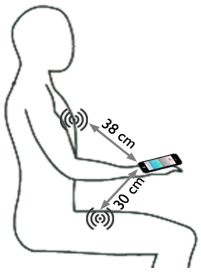

Fig. 8. Near-line-of-sight (nLOS) experimental setup.

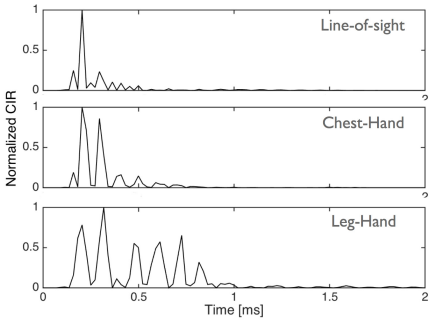

Fig. 9. Ultrasonic in-air CIR for LOS (top), chest-hand nLOS (center) and leg-hand nLOS (bottom).

in Section III-D, and allows users to select which application to run on the wuMote

V. Performance Evaluation

In this Section, we demonstrate the feasibility of ultrasonic communications for wearable devices through experiments, and we evaluate the performance of the U-Wear prototypes discussed in Section IV.

A. PHY Layer Performance

Experiment Setup: The experiment setup consists of a wuMote communicating bidirectionally with a wuMaster in two different scenarios, line-of-sight (LOS) and near-line-of-sight (nLOS). In the LOS scenario the two devices are aligned, $50\ cm$ apart, without obstacles in between, so as to minimize reflections and scattering. In the nLOS scenario, we locate the wuMotes along the body of a user, on the chest and on the right leg, as shown in Fig. 8. The wuMaster, i.e., the smartphone, is held in the user's right hand. Under this setup, objects within the propagation area cause reflections and scattering that introduce ISI and degrade the communication performance. In Fig. 9, we show the uplink CIRs of the three scenarios discussed above. We observe that, in the LOS scenario, the CIR contains a single dominant component. In the nLOS scenario, because of the multipath there are multiple components that contribute to ISI distortion at the receiver. In particular, in the chest-hand setup, the CIR clearly presents a second path, most likely because of a reflection from the user's hand, while in the leg-hand setup we can count up to 6 paths, most likely caused by multiple reflections from the user's trunk and hand. The coherence bandwidth in these three scenarios is approximately $21\ kHz$, $14\ kHz$ and $6\ kHz$, respectively.

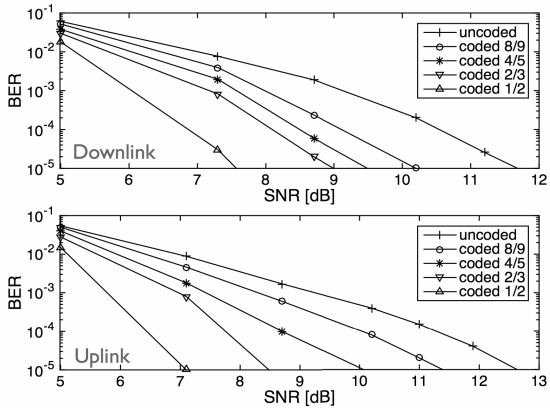

Fig. 10. BER of the downlink (top) and uplink (bottom) in LOS as a function of the SNR for different coding rates.

For each BER measurement we transmit up to 600 packets of 32 bytes, i.e., approximately 256 kilobits, containing pseudorandom-generated raw data. The experiment was performed indoor with a temperature of about 21° C and relative humidity around 30%. We configure the physical layer such that each GMSK symbol is represented by 16 samples. The sampling rate is set to $44.1\ kHz$ as required by the audio hardware in use. Based on this, the raw physical layer data rate, obtained as the ratio between sample rate and sample per symbol, is approximately $2.76\ kbit/s$.

We tune the GMSK BT product to 0.7 to guarantee a good tradeoff between ISI distortion and spectral efficiency. The resulting signal bandwidth is about $2\ kHz$, which is lower than the coherence bandwidth of the three experimental setups, thus complying with the definition of narrowband transmission scheme. The central frequency is set to $18\ kHz$, which, while still in the audible frequency range, represents a good tradeoff between low audibility, fair propagation efficiency, and fair acoustic generation and detection with the COTS microphones and speakers in use. Specifically, we found that $18\ kHz$ is the highest frequency, given the spectral response of microphones and speakers in use, for which we could obtain highly reliable communications, i.e., relatively low BER, in the range of distances of interest, i.e., up to $1\ m$. At the same time, the signal transmission is almost inaudible by the user wearing the device. Finally, given the multipath effect caused by several body reflections, we use a 64-bit PN-sequence as preamble for synchronization and channel estimation.

BER Performance in LOS: In Fig. 10 (top), we show BER results for the downlink, i.e., from the wuMaster to the wuMote, and we compare the performance of an uncoded transmission scheme to four coded transmission schemes with coding rates in {8/9, 4/5, 2/3, 1/2}. The information rate for the five transmission schemes ranges from $2.76\ kbit/s$ for the uncoded transmissions to $2.45\ kbit/s$ for coding rate 8/9, $2.20\ kbit/s$ for coding rate 4/5, $1.84\ kbit/s$ for coding rate 2/4, and $1.38\ kbit/s$ for coding rate 1/2. Figure 10 (bottom) shows the same comparison for the uplink, i.e., from the wuMote to the wuMaster. The SNR is calculated at the receiver as the ratio between the received average signal power and the average noise power measured after amplification and high-pass filtering. We vary the measured

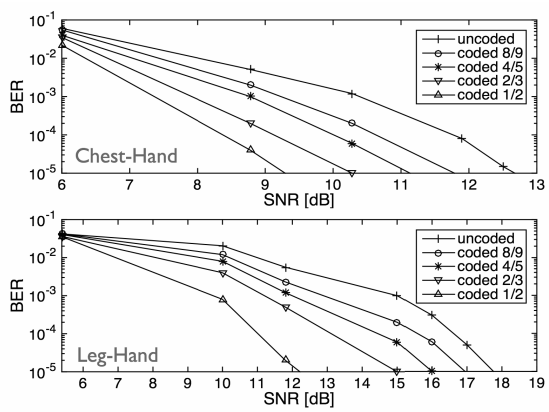

Fig. 11. BER of the chest-hand (top) and of the leg-hand uplinks as a function of the SNR for different coding rates.

SNR by reducing the signal power driving the transmitter speaker. In downlink, we reduce the volume of the smartphone speaker, while in uplink we reduce the signal full-scale at the input of the amplifier.

From Fig. 10, we observe that the BER is a decreasing function of the SNR, and that the FEC scheme mitigates the channel distortion by recovering part of the channel errors. At $5\ dB$ SNR the BER is too high for the FEC to have an impact on the communication performance. Over $5\ dB$ SNR, higher coding rate transmissions have clearly better mitigation performances, thus lower BER.

By measuring the power at the output of the wuMote amplifier, we see how our prototypes achieve $2.76\ kbit/s$ on an uncoded uplink transmission, with a 10^{-5} BER, using a transmission power of $13\ dBm$ ($20\ mW$), i.e., $13\ dB$ SNR at the receiver. We can lower the transmission power by compensating with lower FEC coding rate, thus reducing the information rate. For example, in the current implementation, for a transmission power of $7\ dBm$ ($5\ mW$), i.e., $7\ dB$ SNR, our prototypes achieve $1.38\ kbit/s$ with a 10^{-5} BER using a coding rate of 1/2. Finally, by using a GMSK scheme for the near-ultrasonic frequencies, our prototypes achieve relatively high data rates when compared to previously proposed near-ultrasonic systems; more importantly, it ensures virtually inaudible click-free transmission because of the GMSK phase-continuity as discussed in Section III-A.1.

BER Performance in nLOS: Figure 11 shows the BER performance of uplink transmissions in nLOS scenario chest-hand setup (top) and leg-hand setup (bottom). We observe that, while the curves follow the same pattern as in the LOS scenario, the corresponding BER levels are higher because of the worse channel conditions. The BER in the chest-hand scenario is slightly higher than the LOS one, i.e., about $1\ dB$ more of SNR is required for the same BER. Differently, in the leg-hand scenario we need an increase of $4\ dB$ SNR to achieve the same BER performance of the LOS scenario. In the chest-hand uplink, our prototypes achieve $2.76\ kbit/s$ with a 10^{-5} BER using a transmission power of $19\ dBm$ ($80\ mW$), i.e., about $13\ dB$ SNR at the receiver, while the same BER is obtained with $16\ dBm$ ($45\ mW$) transmission power, i.e., approximately $9\ dB$ SNR at the receiver, halving the data rate through FEC coding. In the leg-hand uplink, we obtain 10^{-5} BER with a transmission power of $24\ dBm$ ($250\ mW$), i.e., about $18\ dB$ SNR at the receiver, for uncoded transmission at $2.76\ kbit/s$ and, and $21\ dBm$ ($130\ mW$) of transmission power, i.e., approximately $12\ dB$ SNR at the receiver, for coded transmission at $1.78\ kbit/s$.

These results show how multipath effect and higher attenuation caused by the user's clothing require higher power transmission as compared to the LOS scenario. Even though ultrasonic signals are further attenuated by solid materials, they can still be used to communicate over short distances through clothing. In general, using speakers and microphones with wider flat bandwidth or custom-made optimized ultrasonic transducers can reduce the required transmission power. In fact, a significant portion of the transmission power is lost during the electro-acoustic conversion in the COTS speaker and microphone in use, which are not designed to operate efficiently at near-ultrasonic frequencies.

B. MAC Layer Performance

We evaluate the performance of the MAC layer protocols implemented on the prototypes, i.e., polling and ALOHA, in terms of data delivery delay as a function of the number of nodes in the network.

Experiment Setup: We set up a M/S configuration where devices lay in nLOS on a 2-dimensional surface, and each wuMote is positioned $40\ cm$ apart from the wuMaster. The experiment consists of collecting data at the wuMaster from up to four wuMotes using polling or ALHOA MAC protocols. We consider six different parameters than can be fetched, and we distribute these among four wuMotes.

Adaptive Polling: Using the polling protocol, the wuMaster fetches data from one node a time. The wuMotes are addressed through physical addresses, e.g., node ID. The PHY layer adaptation allows to reactively adapt the FEC coding rate based on the packet drop rate experienced at the wuMaster to minimize the number of consecutive retransmissions. Specifically, every time the wuMaster retransmits a fetching packet, a lower coding rate is used from the set {8/9, 4/5, 2/3, 1/2}. We fix the maximum number of retransmissions for each fetch command to four. We evaluate the protocol in terms of data delivery delay, which we define as the time between the instant when the first fetching packet is transmitted by the wuMaster and the instant when the last data packet is correctly received at the wuMaster. In Fig. 12 (top), we show the polling data delivery delay as a function of the number of nodes in the network for two levels of SNR measured at the wuMaster, i.e., $10\ dB$ and $14\ dB$, and two coding rates, i.e., 8/9 and 4/5. As expected, since each node in average is granted the same time to transmit, we observe that the delivery delay increases linearly with the number of nodes in the network. Moreover, since retransmissions are only caused by the channel conditions, i.e., there are no collisions among different users, the delivery delay decreases by increasing the SNR or the coding rate. Figure 12 (bottom) shows the delivery delay as a function of the SNR for two fixed coding rates, i.e., 8/9 and 4/5, and for the adaptive scenario. We observe that at lower SNR, a coding rate of 8/9 gives delivery delays higher than a coding rate of 4/5 because of the frequent

TABLE I

HEART-RATE WITH R AND RR METHOD (LEFT). RR INTERVAL MEAN μ AND STD. DEVIATION σ (CENTER). STEP COUNTER (RIGHT)

Trace	6s R[bpm]	6s RR[bpm]	Ref HR[bpm]
16265	100	97	96
16272	60	62	62
16273	100	97	95
16420	90	94	95
16483	100	97	97
16539	80	80	79
16773	70	74	75
16786	70	72	71
16795	70	67	65
17052	70	68	69

Trace	μ[s]	σ[s]	Ref. μ[s]	Ref. σ[s]
16265	0.62	0.62	0.016	0.019
16272	0.96	0.96	0.109	0.115
16273	0.64	0.64	0.023	0.049
16420	0.63	0.63	0.015	0.018
16483	0.62	0.62	0.015	0.012
16539	0.74	0.75	0.062	0.054
16773	0.79	0.79	0.056	0.058
16786	0.85	0.84	0.046	0.036
16795	0.91	0.92	0.070	0.069
17052	0.85	0.85	0.047	0.047

Trace	wuMote	iOS	Real
walk_0	44	49	46
walk_1	39	39	40
walk_2	48	48	50
run_0	32	33	34
run_1	37	42	40
run_2	32	33	32
climb_up	19	19	18
climb_down	17	18	18
climb_do_up	34	34	39

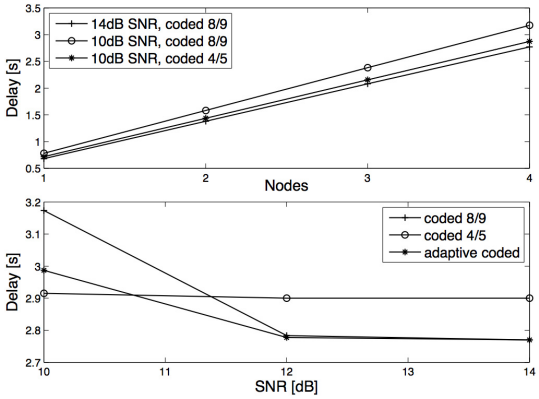

Fig. 12. Polling data delivery delay as a function of number of nodes for different level of SNR and coding rates (top), and as a function of the SNR for non-adaptive and adaptive scenarios.

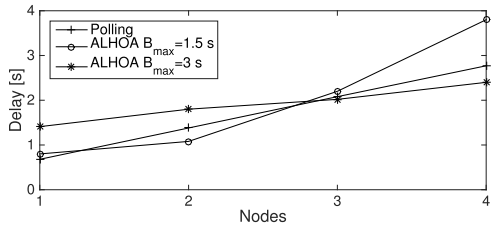

Fig. 13. ALOHA data delivery delay as a function of number of nodes for different B_{max}, compared to polling.

retransmissions due to higher BER at the PHY layer. On the contrary, at higher SNR a coding rate of 4/5 introduces more overhead than needed, giving higher delivery delays than coding rate 8/9. As expected, the adaptive scenario results in delivery delays in between the two fixed coding rate scenarios.

ALOHA: With ALOHA, we use the content-centric addressing scheme. Hence, the wuMaster broadcasts a request message to fetch data from multiple wuMotes. The wuMotes transmit the requested data, if available, by accessing the channel randomly. Finally, we select the backoff time between transmissions from 0 to a maximum backoff B_{max}, and we vary it during our experiments, while fixing the SNR to $14\ dB$ and FEC coding rate to 8/9. Figure 13 shows the data delivery delay as a function of the number of nodes in the network for two different values of B_{max}, i.e., $1.5\ s$ and $3\ s$. We compare the results with the data delivery delay experienced by the polling protocol for $14\ dB$ SNR and 8/9 coding rate. When the number of nodes in the network is lower than three, we observe that $B_{max} = 1.5\ s$ gives lower delay than $B_{max} = 3\ s$. Here, a higher B_{max} increases the probability of selecting a higher backoff time, leading to channel underutilization. On the other hand, for number of nodes higher and equal to three, $B_{max} = 1.5\ s$ gives high probability of collisions, thus higher delay due to retransmissions.

C. Data Processing Performance

We evaluate the data processing accuracy in terms of displacement between the obtained outputs, i.e., what the application reads on a given sensor trace, and the expected ones, i.e., what the application should read on that given sensor trace. We consider three applications: i) heart rate monitor, ii) ECG *RR interval* monitor, and iii) step counter.

ECG Processing: To show a fair comparison, we feed the ECG-based applications with reference raw sensor data from the MIT-BIH Database [37]. The traces are sampled at $128\ Hz$. We extract, and load to the wuMote, 10 one-minute long traces from the MIT-BIH recordings. In Table I (left), we show the heart rate estimation of the wuMote using the *R method*, second column, and *RR method*, third column, discussed in Section IV-A.2, and we compare these with the heart rate reference provided by the MIT-BIH database, fourth column. The first column shows the database trace ID. We observe that both *R method* and *RR method* give a good estimate of the reference heart rate, offering an average accuracy of 96.1% and 98.7%, respectively.

In Table I (center), we show the *RR interval* mean μ and standard deviation σ estimated by the wuMote, second and fourth columns, and we compare these with the *RR interval* reference statistics provided by the MIT-BIH database, third and fifth columns. We observe that the wuMote accurately estimates the *RR interval* mean, i.e., around 99.6% of accuracy. For the standard deviation σ we obtain lower accuracy, i.e., 83.6%, for two reasons, (i) the relatively low sampling rate gives a sensibility of $8\ ms$, which can affect the measurement of small quantities such as the standard deviation, (ii) failures in the peak finding algorithm also affect the measurement. Higher sampling rate and outlier detection techniques could be used to further enhance the standard deviation measurement.

Accelerometer Processing: We record ten 3-dimensional accelerometer traces with a sample rate of $60\ Hz$ using Sensor Log for iOS. In Table I (right), we show the footstep count estimated by the wuMote, second column, and we compare this with the footstep estimate of the iOS device, third column, and real footstep number counted by the user while performing the steps, fourth column. The first column shows the trace name, where we list 3 walking traces, 3 running traces and 3 stair climbing traces, i.e., downward, upwards and down/upwards. We observe that, in average, the wuMote counts footsteps with the same accuracy of the iOS device,

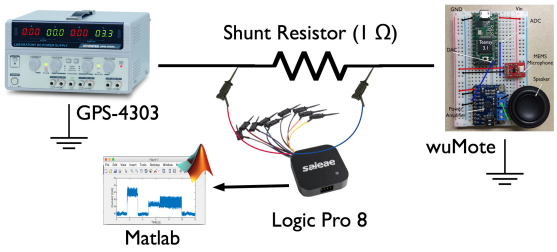

Fig. 14. Diagram of the current measurement setup (not in scale).

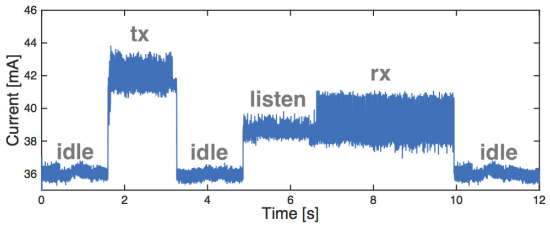

Fig. 15. Measured current drawn by the wuMote.

i.e., approximately 94% with respect to the number of steps counted by the user.

D. Power Consumption Performance

We evaluate the power consumption of the wuMote prototype in terms of current drawn from an external power supply in different states, e.g., idle, transmitting or receiving.

Measurement Method: The measurement is performed using the shunt resistor method, which consists of measuring the voltage drop along a small resistor, i.e., the shunt resistor, connected in series between the power supply and the load, i.e., the wuMote. By dividing the voltage drop by the value of the shunt resistor, $I = V/R$, we obtain the current flowing through the resistor, thus the current drawn by the wuMote.

Figure 14 shows a diagram of the measurement setup. In this setup, the voltage drop is measured using two analog inputs of the Saleae Logic Pro 8 logic analyzer to capture voltages at the two ends of a 1 Ω shunt resistor. The voltage measures, sampled at 12.5 MHz, are saved on a host computer and exported to Matlab for processing; the voltage drop is obtained as the difference between the two voltage measures, and the current flowing through the 1 Ω shunt resistor is equal to the voltage difference. Power is supplied by a 3.3 V DC power, Instek GPS-4303.

Measurement Results: We consider a scenario where the wuMote switches between different processing states, i.e., *deep sleep*, *run idle*, *run tx*, *run listen* and *run rx*. During these tests, the power amplifier is in shutdown mode (consuming less than 2 μA), unless the wuMote is transmitting. The microphone and preamplifier are always on, consuming less than 250 μA.

Figure 15 shows the measured current drawn by the wuMote as a function of time for the four run modes over 12 s recording. This includes both the current drawn by Teensy as well as the current drawn by the communication unit, i.e., power amplifier, preamplifier, speaker and microphone. In *deep sleep* mode, Teensy peripherals are not clocked, and the device can be awakened using a wakeup interrupt. In this state, the wuMote current consumption is as low as 450 μA, which includes both Teensy and the mic/preamp current drawn. In *run idle* mode, Teensy's peripherals are clocked and running at 96 MHz, and current consumption goes up to 36 mA. In *run tx* mode the device requires 41.8 mA to transmit a packet assuming 13 dBm (20 mW) transmission power, which includes Teensy processing 36.7 mA, and the power amplification 7.1 mA consumption. In *run listen* and *run rx* mode, the device is waiting for a packet and receiving a packet, and it requires 38.5 mA and 39 mA, respectively. Table II summarizes the average current and power consumption measured in each state. Power is obtained multiplying the current drawn for the supply voltage, i.e., $P = VI$.

TABLE II
AVERAGE CONSUMPTION OF THE WUMOTE FOR DIFFERENT STATES

State	Current [mA]	Power [mW]
deep sleep	0.45	1.48
run idle	36	118.8
run tx	41.8	137.9
run idle	38.5	127
run rx	39	128.7

Most of the power consumption of the wuMote is due to the Teensy microcontroller running at full clock speed. However, teensy run idle power consumption can be substantially reduced by moving into a *low power run mode* that reduces the current drawn to about 2 mA, and can be used when no tx/rx operations are performed, e.g., during MAC, network and application layer processing. Similarly, power consumption during tx/rx operations could be decreased by further optimizing the code to shut down any peripheral that is not in use in those states. For example, the DAC could be off when receiving and the ADC could be off when transmitting. This optimization is however lower bounded by the idle Teensy current consumption, i.e., the current drawn when running a blank project with no active peripherals, which is around 25 mA at 96 MHz clock.

To further reduce the power consumption, as well as the size of a future design of the wuMote, the Cortex-M4 embedded on Teensy could be replaced with a low-power microcontroller such as the mm-size low-power Freescale KL03 Cortex-M0 microcontroller. Using the KL03 the tx/rx current consumption could go down to around 10 mA, and the deep-sleep current drawn could become as small as 70 nA, paying the price of more stringent constraints because of the very limited resources available, e.g., 2K of RAM and 8K of Flash.

Based on the measurement performed, it is clear that, the current wuMote design is consuming more energy to operate when compared to commercially available RF-based devices. For example, the BLE TI CC2540 and CC2640 chips, consume as low as 26 mA and 6.1 mA, respectively when transmitting data. Moreover, BLE offers much higher throughput, i.e., between 5 and 100 $kbit/s$, than the current wuMote prototype, i.e., between 1 and 2 $kbit/s$, therefore leading to much lower energy-per-bit consumption.

However, as discussed in the introduction, the current performance of low-power commercial RF systems is the result of many years of research and development in multiple fields and of a multi-billion dollar industry. In this paper we present a proof of concept and explore several system tradeoffs based on

a prototype built entirely using COTS components, therefore not optimized for low-energy operations. Future designs that use piezoelectric ultrasonic transducers instead of speaker and microphone, and lower-power microcontrollers with more efficient DSP operations, could substantially reduce the energy consumption of the device, therefore making it competitive with commercially-available RF-based devices.

VI. Related Work

The idea of using ultrasonic waves for in-air communications has been proposed in the past, and several studies have successfully proven the viability of using ultrasonic waves as an alternative to RF waves for short-range and medium-range in-air communications. In [38], the authors discuss the possibility of using sounds for in-air communication purposes, and explore several different physical-layer approaches that can be used in a ubiquitous computing environment. In [50] and [51], the authors studied the performance of ultrasonic in-air transmissions over short-range links of a few meters using custom-made transducers. The proposed solutions achieve data rates in the order of $100\ kbit/s$, on relatively large bandwidths, i.e., $80\ kHz$ and $56\ kHz$, using QPSK and on-off-keying (OOK). In [41], medium-range communications, i.e., up to $20\ m$, are achieved with data rates up to $100\ bit/s$, on a $4\ kHz$ FSK-modulated bandwidth. While U-Wear is based on the same idea of using ultrasonic waves for in-air communications, the framework goes well beyond a simple point-to-point transmission scheme, such as those presented in the above works. In this paper, to the best of our knowledge, we propose for the first time the use of ultrasonic waves for interconnecting wearable devices in a network, and we present the first networking framework based on ultrasonic communications that offers multi-layer functionalities spanning the PHY, data link, network and application layer.

In the last few years, researchers have proposed to use acoustic waves in the audible and near-ultrasonic frequency ranges to enable in-air communications between devices equipped with COTS microphones and speakers. In [53] and [54], the authors use audible sound around $8\ kHz$ for near-field-communications between smartphones, using OFDM and FSK schemes, and achieving data rates in the order of a few $kbit/s$. In [21] and [55], the authors propose in-air communication systems that operate in the near-ultrasonic frequency range, and achieve low data rates, i.e., up to $20\ bit/s$, over medium-range directional links, i.e., up to $20\ m$. In [45], the authors present techniques to hide data into sound tracks to deliver information to smartphones and to leverage the channel diversity among different users to collaboratively correct transmissions errors. In [46], the authors leverage the mechanisms of bone conduction transmission, to enable secure high-frequency acoustic communications between a smartwatch and a smartphone. While U-Wear can certainly operate in the near-ultrasonic frequency range to enable communications with commercial devices such as smartphones and laptops, among others, it has been primarily designed to provide connectivity between wearable devices at higher ultrasonic frequency ranges. In this paper, we implement near-ultrasonic communications to demonstrate interoperability of the wuMote with commercial smartphones, and also because of the availability of inexpensive COTS audio speakers and microphones that can be used as a proxy for air-coupled ultrasonic transducers. With this idea in mind, we proposed for the first time the use of a narrowband GMSK modulation for the near-ultrasonic frequency range, which enables relatively high data rates when compared to other existing works, i.e., up to $2.76\ kbit/s$ on a $2\ kHz$ bandwidth around $18\ kHz$, and ensures virtually inaudible click-free transmission because of the phase-continuity of the GMSK waveform. Higher data rates can be achieved by using microphones and speakers with wider bandwidth, or by developing custom-made air-coupled transducers. For example, by using OFDM signaling schemes with high-order modulations, e.g., 16-PSK or 32-QAM, on a $24\ kHz$ bandwidth centered around $100\ kHz$, we could possibly achieve data rates in the order of hundreds of $kbit/s$.

VII. Conclusions

In this paper, we presented U-Wear, the first networking framework for wearable medical devices based on ultrasonic communications. We designed two prototypes that implement the U-Wear framework and operate in the near-ultrasonic frequency range, using commercial-off-the-shelf (COTS) speakers and microphones. Despite the limited bandwidth, i.e., about $2\ kHz$, and the COTS audio hardware components not optimized for operating at high frequency, we showed how our prototypes (i) can operate at data rate up to $2.76\ kbit/s$ with bit-error-rate (BER) lower than 10^{-5}, using a transmission power of $20\ mW$; (ii) enable multiple nodes to share the same medium offering tradeoffs between packet delivery delay and packet drop rate; (iii) can implement reconfigurable data processing applications that can extract medical parameter from sensor traces with high accuracy. U-Wear can offer higher performance through specialized hardware components. Future smart devices with wider-bandwidth speakers and microphones will enable higher data rates and lower energy consumption. Through custom-made ultrasonic air-coupled transducers that operate at higher frequency ranges and larger bandwidth, wuMotes may be able to achieve data rates in the order of hundreds of $kbit/s$ with lower energy consumptions.

References

[1] G. E. Santagati and T. Melodia, "U-Wear: Software-defined ultrasonic networking for wearable devices," in *Proc. Annu. Int. Conf. Mobile Syst. Appl. Services (MOBISYS)*, Florence, Italy, May 2015, pp. 241–256.

[2] G.-Z. Yang, *Body Sensor Networks*. New York, NY, USA: Springer, 2006.

[3] J. J. Oresko *et al.*, "A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 3, pp. 734–740, May 2010.

[4] N. Roxhed, B. Samel, L. Nordquist, P. Griss, and G. Stemme, "Painless drug delivery through microneedle-based transdermal patches featuring active infusion," *IEEE Trans. Biomed. Eng.*, vol. 55, no. 3, pp. 1063–1071, Mar. 2008.

[5] S. Patel *et al.*,"Tracking motor recovery in stroke survivors undergoing rehabilitation using wearable technology," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Buenos Aires, Argentina, Aug. 2010, pp. 6858–6861.

[6] A. Salarian *et al.*, "Gait assessment in Parkinson's disease: Toward an ambulatory system for long-term monitoring," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 8, pp. 1434–1443, Aug. 2004.

[7] M. M. Ohayon and C. M. Shapiro, "Sleep disturbances and psychiatric disorders associated with posttraumatic stress disorder in the general population," *Comprehensive Psychiatry*, vol. 41, no. 6, pp. 469–478, Nov. 2000.
[8] *Fitbit Fitness Trackers*, accessed on Oct. 21, 2016. [Online]. Available: https://www.fitbit.com/
[9] *Hexoskin, Wearable Body Metrics*, accessed on Oct. 21, 2016. [Online]. Available: http://www.hexoskin.com/
[10] *APDM, Movement Monitoring Solutions*, accessed on Oct. 21, 2016. [Online]. Available: http://www.apdm.com/
[11] H. S. Berger and H. M. Gibson, "Managing your hospital RF spectrum," *Biomed. Instrum. Technol.*, vol. 47, no. 3, pp. 193–197, 2013.
[12] *FDA, Wireless Medical Devices*, accessed on Oct. 21, 2016. [Online]. Available: http://www.fda.gov/MedicalDevices/DigitalHealth/WirelessMedicalDevices
[13] T. Melodia, H. Kulhandjian, L. Kuo, and E. Demirors, "Advances in underwater acoustic networking," in *Mobile Ad Hoc Networking: Cutting Edge Directions*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds., 2nd ed. Hoboken, NJ, USA: Wiley, 2013, pp. 804–852.
[14] P. Lazik and A. Rowe, "Indoor pseudo-ranging of mobile devices using ultrasonic," in *Proc. ACM Conf. Embedded Netw. Sens. Syst. (SENSYS)*, 2012, pp. 99–112.
[15] R. J. Przybyla *et al.*, "3D ultrasonic gesture recognition," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, Feb. 2014, pp. 210–211.
[16] F. L. Thurstone and H. E. Melton, "Biomedical ultrasonics," *IEEE Trans. Ind. Electron. Control Instrum.*, vol. IECI-17, no. 2, pp. 167–172, Apr. 1970.
[17] G. E. Santagati, T. Melodia, L. Galluccio, and S. Palazzo, "Ultrasonic networking for e-health applications," *IEEE Wireless Commun.*, vol. 20, no. 4, pp. 74–81, Aug. 2013.
[18] G. E. Santagati, T. Melodia, L. Galluccio, and S. Palazzo, "Medium access control and rate adaptation for ultrasonic intra-body sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1121–1134, Aug. 2015.
[19] G. E. Santagati and T. Melodia, "Sonar inside your body: Prototyping ultrasonic intra-body sensor networks," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 2679–2687.
[20] H. Lee, T. H. Kim, J. W. Choi, and S. Choi, "Chirp signal-based aerial acoustic communication for smart devices," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Hong Kong, Apr. 2015, pp. 2407–2415.
[21] L. B. Evans, H. E. Bass, and L. C. Sutherland, "Atmospheric absorption of sound: Theoretical predictions," *J. Acoust. Soc. Amer.*, vol. 51, no. 5B, pp. 1565–1575, 1972.
[22] *Attenuation of Sound Duing Propagation Outdoors—Part 1: Calculation of the Absorption of Sound by the Atmosphere*, document. ISO 9613-1, Dec. 2010.
[23] K. Murota and K. Hirade, "GMSK modulation for digital mobile radio telephony," *IEEE Trans. Commun.*, vol. COM-29, no. 7, pp. 1044–1050, Jul. 1981.
[24] E. Demirors, G. Sklivanitis, G. E. Santagati, T. Melodia, and S. N. Batalama, "Design of software-defined underwater acoustic modem with real-time physical layer adaptation capabilities," in *Proc. Int. Conf. Underwater Netw. Syst. (WUWNet)*, Rome, Italy, Nov. 2014, pp. 1–25.
[25] F. Tufvesson, O. Edfors, and M. Faulkner, "Time and frequency synchronization for OFDM using PN-sequence preambles," in *Proc. IEEE Veh. Technol. Conf.*, Houston, TX, USA, Sep. 1999, pp. 2203–2207.
[26] M. Skolnik, "Radar Handbook," in *Electronics Electrical Engineering*, 3rd ed. New York, NY, USA: McGraw-Hill, 2008.
[27] R. J. Polge and E. M. Mitchell, "Impulse response determination by cross correlation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-6, no. 1, pp. 91–97, Jan. 1970.
[28] J. Proakis and M. Salehi, *Digital Communications*. New York, NY, USA: McGraw-Hill, 2007.
[29] N. Abramson, "The ALOHA system: Another alternative for computer communications," in *Proc. Fall Joint Comput. Conf. (FJCC)*, Houston, TX, USA, Nov. 1970, pp. 281–285.
[30] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part I—Carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. Commun.*, vol. COM-23, no. 12, pp. 1400–1416, Dec. 1975.
[31] Y. Sun and T. Melodia, "The Internet underwater: An IP-compatible protocol stack for commercial undersea modems," in *Proc. ACM Int. Conf. UnderWater Netw. Syst. (WUWNet)*, Kaohsiung, Taiwan, Nov. 2013, pp. 1–37.
[32] *Teensy USB Development Board*, accessed on Oct. 21, 2016. [Online]. Available: https://www.pjrc.com/teensy/
[33] C. L. Chang, K. P. Lin, T. H. Tao, T. Kao, and W. H. Chang, "Validation of automated arrhythmia detection for holter ECG," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Oct. 1998, pp. 101–103.
[34] *vDSP Library, iOS Accelerate Framework*, accessed on Oct. 21, 2016. [Online]. Available: http://goo.gl/6gvpFT
[35] *Novocaine*, accessed on Oct. 21, 2016. [Online]. Available: https://github.com/alexbw/novocaine
[36] *WIT.AI, Natural Language for the Internet of Things*, accessed on Oct. 21, 2016. [Online]. Available: https://wit.ai
[37] *MIT-BIH Normal Sinus Rhythm Database*, accessed on Oct. 21, 2016. [Online]. Available: http://goo.gl/B70gqI
[38] C. V. Lopes and P. M. Q. Aguiar, "Acoustic modems for ubiquitous computing," *IEEE Pervas. Comput.*, vol. 2, no. 3, pp. 62–71, Jul. 2003.
[39] C. Li, D. A. Hutchins, and R. J. Green, "Short-range ultrasonic communications in air using quadrature modulation," *IEEE Trans. Ultrason., Ferroelect., Freq. Control*, vol. 56, no. 10, pp. 2060–2072, Oct. 2009.
[40] W. Jiang and W. M. D. Wright, "Ultrasonic wireless communication in air using OFDM-OOK modulation," in *Proc. IEEE Int. Ultrason. Symp. (IUS)*, Chicago, IL, USA, Sep. 2014, pp. 1025–1028.
[41] S. Holm, O. B. Hovind, S. Rostad, and R. Holm, "Indoors data communications using airborne ultrasound," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Philadelphia, PA, USA, Mar. 2005, pp. 957–960.
[42] R. Nandakumar, K. K. Chintalapudi, V. Padmanabhan, and R. Venkatesan, "Dhwani: Secure peer-to-peer acoustic NFC," in *Proc. ACM Conf. SIGCOMM*, Hong Kong, 2013, pp. 63–74.
[43] B. Zhang *et al.*, "Priwhisper: Enabling keyless secure acoustic communication for smartphones," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 33–45, Feb. 2014.
[44] M. Hanspach and M. Goetz, "On covert acoustical mesh networks in air," *J. Commun.*, vol. 8, no. 11, pp. 758–767, 2013.
[45] R. Frigg, G. Corbellini, S. Mangold, and T. R. Gross, "Acoustic data transmission to collaborating smartphones an experimental study," in *Proc. 11th Annu. Conf. Wireless-Demand Netw. Syst. Services (WONS)*, Apr. 2014, pp. 17–24.
[46] S.-C. Kim and S.-C. Lim, "Transferring data from smartwatch to smartphone through mechanical wave propagation," *Sensors*, vol. 15, no. 9, p. 21394, 2015.

G. Enrico Santagati (S'13) received the B.S. and M.S. degrees in telecommunication engineering from the University of Catania, Catania, Italy, in 2010 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. He is also with the Wireless Networks and Embedded Systems Laboratory, Northeastern University, under the guidance of Prof. T. Melodia. His current research interests are in ultrasonic intrabody networks and software-defined radios.

Tommaso Melodia (M'07–SM'16) received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2007. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. His research has been supported by the National Science Foundation, Air Force Research Laboratory, and the Office of Naval Research, among others. His current research interests are in the modeling, optimization, and experimental evaluation of networked communication systems, with applications to ultrasonic intrabody networks, cognitive and cooperative networks, multimedia sensor networks, and underwater networks. He was a recipient of the National Science Foundation CAREER Award and coauthored a paper that was recognized as the ISI Fast Breaking Paper in the field of computer science in 2009. He was a recipient the Best Paper Awards of the ACM WUWNet in 2013 and 2015. He was the Technical Program Committee Vice Chair of the IEEE GLOBECOM 2013. He was the Technical Program Committee Vice Chair of the IEEE INFOCOM 2013 for information systems. He serves on the editorial boards of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON MULTIMEDIA, and *Computer Networks*.