Internet-of-Microchips: Direct Radio-to-Bus Communication with SPI Backscatter

Songfan Li
University of Electronic Science and Technology of China
songfanli@std.uestc.edu.cn

Chong Zhang
University of Electronic Science and Technology of China
zhangchong@std.uestc.edu.cn

Yihang Song
University of Electronic Science and Technology of China
songyihang@std.uestc.edu.cn

Hui Zheng
University of Electronic Science and Technology of China
daystzh@gmail.com

Lu Liu
University of Electronic Science and Technology of China
liulu_nise@std.uestc.edu.cn

Li Lu
University of Electronic Science and Technology of China
luli2009@uestc.edu.cn

Mo Li
Nanyang Technological University
limo@ntu.edu.sg

ABSTRACT

Energy consumption of Internet-of-Things end devices is a major constraint that limits their long-term and large-scale deployment. Conventionally, the radios and processors used in these end devices are major power consumption that drains at the level of milliwatts (*m*Ws). However, in recent decades, backscatter communication has dramatically reduced the power consumed by the radios in end devices to microwatts (μWs), and thus the processor remains the major bottleneck for energy optimization.

In this paper, we propose a processor-free architecture as a novel design that allows the radio to interface directly with peripheral sensor chips for control and data collection, thereby separating the processors from the end device design to significantly reduce the energy consumed by end devices. The main problem is that the peripheral chips are designed to be accessed by the processor via a standard digital bus and they cannot communicate directly with the radio. In order to support such processor-free design, we propose *radio-to-bus* (R2B) as a novel communication paradigm that allows direct data exchange between a backscatter radio and the serial peripheral interface (SPI) bus. We implement the processor-free architecture in proof-of-concept prototypes and demonstrate that the power consumption decreases by 4.5 times compared with the conventional end device design.

CCS CONCEPTS

• **Networks** → *Home networks*; • **Hardware** → **Wireless devices**; • **Computer systems organization** → **Embedded systems**.

KEYWORDS

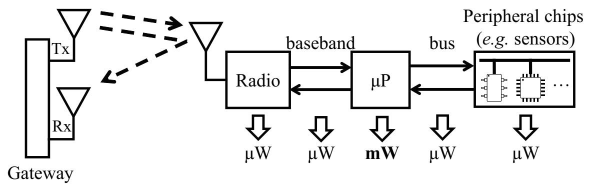
backscatter, Internet of things, SPI

ACM Reference Format:
Songfan Li, Chong Zhang, Yihang Song, Hui Zheng, Lu Liu, Li Lu, and Mo Li. 2020. Internet-of-Microchips: Direct Radio-to-Bus Communication with SPI Backscatter. In *MobiCom 2020 (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3419182

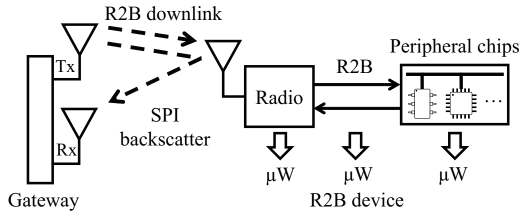
1 INTRODUCTION

Energy consumption of end devices is a major constraint that limits the long-term and large-scale deployment of Internet-of-Things (IoT). The conventional architecture for IoT end devices requires three fundamental components comprising the radio for information exchange with the gateway, peripheral sensors for sensing the physical world conditions, and a microprocessor (μP) that interfaces between the radio and the peripherals, and controls them. While various peripheral sensors are triggered infrequently and usually operate at the level of microwatts (μWs), the radio and microprocessor are major power consumption that drains at the level of milliwatts (*m*Ws). In recent decades, many methods have been proposed for reducing the power consumed by the radio, where the most recent advance in backscatter communication allows energy harvesting during Tx/Rx to dramatically reduce the power profile of the radio to μWs, and thus the microprocessor remains as the major bottleneck for energy optimization.

We may tune the microprocessor into deep sleep whenever possible but the frequent interfacing operations between the radio and peripherals must pass through the microprocessor, and thus its sleep is interrupted and power spikes occur during mode switches. In typical use cases, general purpose microprocessors such as the ARM/MSP430 family still operate at the order of *m*Ws [16] on average, thereby reducing the lifetimes of end devices to less than a year. Alternatively, we may customize the processor design by using an application-specific integrated circuit (ASIC) for a particular use and to greatly reduce the power consumption. However, the purposes of IoT end devices often change in various applications, and thus the design of the ASIC-based processor has to be modified



(a) Conventional architecture of backscatter devices.



(b) R2B enabled Internet-of-Microchips architecture.

Figure 1: Radio-to-bus (R2B) communication allows the radio to directly control the peripherals, thereby facilitating the processor-free design of IoT digital end devices with lower power consumption as well as eliminating the need for embedded programming.

to interface diverse sensors and perform different data processing roles, which is not practical due to the high costs involved.

In this paper, we rethink the conventional IoT end device design and ask the following question: *Can we allow the radio to directly interface with the peripheral sensor chips, thereby shifting data processing and sensor control to the gateway?* If this can be achieved, then we can remove the microprocessor from the design and envision a future Internet-of-Microchips architecture where IoT gateways have direct wireless access to sensor chips deployed for control and data collection purposes, which would allow high deployment flexibility and significantly reduce the energy consumption of end devices. The key challenge with a processor-free solution is that embedded designs are currently used with a microprocessor in between the radio and peripherals, where the baseband signals are first digitized and a standard digital bus like SPI or I^2C is defined to carry data and allow control of the peripheral chips.

In order to support the processor-free design of end devices, we propose *radio-to-bus* (R2B) as a novel communication paradigm that allows direct data exchange between a backscatter radio and the peripheral bus. Based on R2B communication, power-hungry computations and controls are shifted to the gateway. Hardware-specific programming can be eliminated, and all of the controls and operations are executed through remote wireless communication. Figure 1(b) illustrates the R2B design. We connect the sensor chips via an SPI to a backscatter radio on each end device. On the downlink, the gateway (Tx) sends bus-compatible baseband signals that simulate bus operations. The signals can be demodulated by the end device and conveyed directly through the SPI to interact with sensor chips. On the uplink, the end device backscatters interrogation signals from the gateway while also modulating the digital data from the SPI to deliver it directly to the gateway (Rx). This design maintains end-to-end communication with μW level power efficiency. In order to implement this design, we address the following three technical obstacles.

First, on the downlink, R2B communication requires the direct conversion of baseband signals into SPI bus-compatible signals without a processor. The baseband data comes in streams via one wire, but the SPI bus (and other general purpose serial interfaces such as I^2C) requires at least two wires simultaneously for both the data and a clock reference to allow synchronous data exchange. Rigorous specifications must be followed, including symbol definitions, synchronization, and protocol controls. In this paper, we propose a hardware/software co-designed demodulation scheme that produces SPI compatible data as well as an SPI compatible clock from the received baseband signals. We implement our design based on pulse-interval-encoding (PIE), which a commonly used data coding scheme for downlink transmissions.

Second, in order to read data from the peripheral chips, the conventional embedded designs allow the microprocessor to generate a reference clock to drive the peripherals to output the desired data on the SPI bus. Thus, without the microprocessor, we require a substitute to produce the clocks. Intuitively, we may employ a local clock generator but this method leads to subsequent problems, including local clock control in the absence of the processor. In R2B communication, we leverage the gateway to provide an external clock to end devices over the air. In particular, the gateway performs a downlink transmission that contains an SPI compatible reference clock. After demodulation, the peripheral chips receive the external clock and generate SPI bus data, which are then transmitted on the uplink to the gateway.

Third, the SPI bus data are described by both the data signals and the reference clock, so the backscatter communication on the R2B uplink needs to incorporate both the data and clock in its modulation in order to allow the gateway to successfully decode the symbols in a synchronized manner. However, the existing backscatter techniques focus on delivering the baseband signal and they barely satisfy the dual-signal transmission requirement imposed by the SPI bus specifications. In R2B communication, we leverage the solution to the second obstacle to help us tackle this problem. The external clock signal on the downlink is utilized to transfer the data output from the SPI, so the end device backscatters the external clock signal and simultaneously modulates the SPI output data signal. In this manner, we transform the dual-signal transmission problem into single channel delivery (only data) and we can use the existing backscatter techniques to achieve R2B uplink transmissions.

To test the feasibility of our design, we build two proof-of-concept prototypes, where one is implemented with discrete off-the-shelf components to demonstrate the processor-free concept and the other is an FPGA implementation to quantify the power consumption. In addition, we implement the gateway with software-defined radio (USRP 2922). We evaluate the prototypes in comprehensive experiments. The results show that R2B can achieve the data rate up to 200 kbps for both the downlink and uplink. The experiments also show that the practical operating distance for R2B backscatter communication can reach up to 30 m. When operating an accelerometer at 400 Hz, we show that the R2B prototype consumes 25.9 μW of power, which is 4.5 times lower than WISP5.0 that stands for the conventional processor-based design. We also

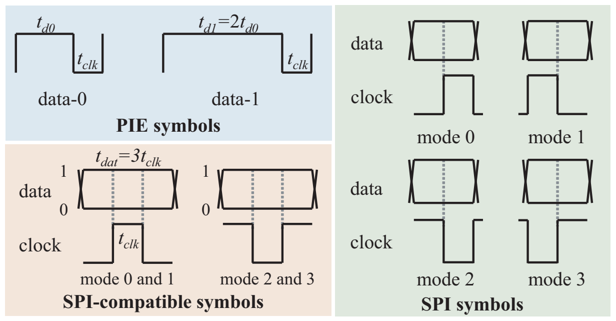


Figure 2: Symbols for PIE and SPI data. The dashed line indicates the clock edge for receiving SPI data.

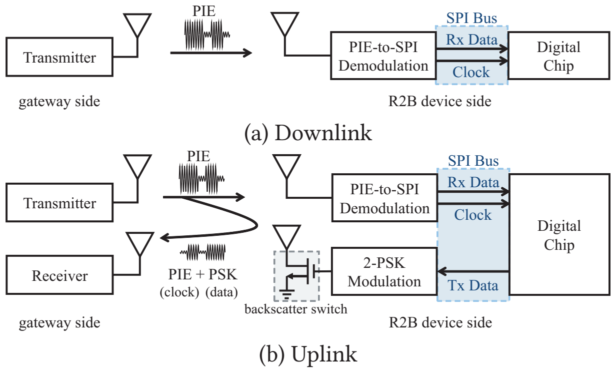


Figure 3: Overview of Radio-to-Bus communication.

compare R2B with EkhoNet, which is a custom FPGA-based design for backscatter sensor platforms, and observe that R2B reduces the power consumption by 26%. In addition to the power benefits, we demonstrate the potential of R2B for eliminating the need for embedded programming in the end device and for supporting plug-and-play sensors.

2 A PRIMER FOR SPI BUS

In our prototype, peripheral chips are connected on the SPI bus for R2B communication. In the following, we provide some background details of SPI to better understand our design.

SPI Bus Configuration: The SPI bus imposes a master-slave configuration where a unique master controls all slaves on the bus. In the conventional architecture, an onboard microprocessor has the master role. In contrast, gateways play the master role via wireless communication in R2B.

SPI Signal Interface: The SPI bus consists of four types of logic signals comprising input data (into chip), output data (from chip), clock, and chip select. For both the input and output, each bit of data is driven by the clock signal. In particular, a clock pulse contains two clock edges, where one is used to receive a bit from the input data signal and the other edge is used for transmitting a bit of data at the output. In addition, chip select is used to support selection by the master in multi-chip communication. For convenience, we first focus on the single chip case where only one sensor is on the bus. We then discuss how to approach the multi-chip case in § 6.

SPI Data Symbols: SPI symbols are described by both the data and clock. On data lines, the logical high voltage represents bit 1, and vice versa. As shown in Fig. 2 (right), the SPI bus has four symbol modes according to the clock polarity and phase. First, the clock polarity indicates the signal in idle. For example, the clock signal in mode 0 and 1 idles at logical low, and it comprises a logical high clock pulse. Second, the clock phase represents the timing of the data relative to the clock edges. In particular, in mode 0 and 2, the input data are sampled on the leading clock edge. Furthermore, the output data occurs on the next clock edge. In our prototype, in order to simplify the *downlink* receiver circuit in R2B devices, we adopt the SPI-compatible pattern on the downlink. As shown in Fig. 2 (bottom-left), the four modes in SPI can be transformed into a compatible pattern. The data signal is prolonged to cover a whole clock cycle so both clock edges can be used to receive data. As a result, only two modes exist in this pattern and they are determined only according to the clock polarity.

3 R2B IN A NUTSHELL

3.1 Downlink

On the downlink, our goal is to achieve data transmissions from the gateway to the peripheral chips inside R2B devices. According to our description of the SPI above, the digital chips simultaneously require input data and clock. Thus, we attempt to provide both the data and clock information over the air on the downlink.

We provide an overview of downlink communication in Fig. 3(a). We leverage PIE to stitch the data and clock together on the downlink transmissions. The symbol for PIE data is shown in Fig. 2 (top-left). The length of a logical high period represents the data value. Each symbol contains a logical low pulse that includes the clock information. After the R2B device receives the downlink transmission, the PIE data is detected by a radio frequency (RF) envelope detector [3] and PIE is treated as the baseband on the downlink. Furthermore, the PIE baseband is directly converted into SPI-compatible symbols. The technical design is explained in § 4.

3.2 Uplink

In the following, we discuss how to resolve the following two problems on the uplink.

First, how can the peripherals generate output data without a processor to provide the clock? We tackle this problem by leveraging a downlink transmission signal as the external clock source. The downlink transmission is encoded in PIE format and it provides the clock to transfer the data output. The SPI bus supports communication in full duplex mode, so the chips can output data while simultaneously receiving data from the input. Thus, the downlink design in R2B devices does not need any extra modifications. For only uplink communication, the downlink transmission can be all "data-0" or "data-1" to only deliver clock information.

Second, how can we simultaneously transmit bus data and clock using backscatter? The downlink transmission can also help us to resolve the second problem. At a high level, when the external clock enters the chip, the output data signal is modulated and further operates the backscatter switch to reflect the downlink signal. The downlink RF signal comprises the clock information

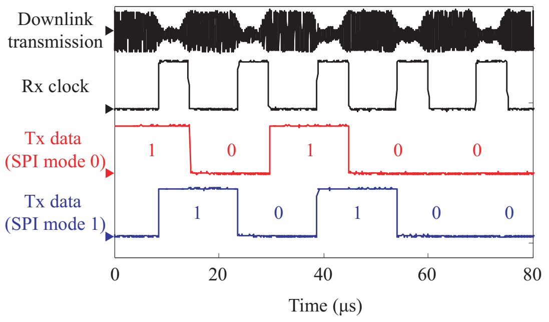


Figure 4: The high amplitudes of the downlink signal are used to carry the bus output data. The output data for SPI mode 2 (3) are the same as mode 0 (1).

that is leveraged to generate the output data from the peripherals. Therefore, the backscattered signal contains both data and clock information, and it is decoded correctly on the receiver side. We summarize the uplink communication design in Fig. 3(b).

The following important question still remains: Is it possible to re-modulate the output data on the ASK-modulated downlink RF signal? The low amplitude period in the ASK signal has weak strength. If the bus output data is modulated during the low amplitude period, then the signal received by the receiver may become too weak to be demodulated.

We achieve bus backscatter transmission according to an observation. As shown in Fig. 4, the SPI output signal is stable between two consecutive clocks. In particular, SPI-interfaced chips produce a bit on a clock edge and the output is maintained until the next transmission clock edge. The mean time between two transmission edges covers the high amplitude of the downlink RF signal. Given that ASK modulation on the downlink does not change the signal phase, we achieve binary phase shift keying (2-PSK) modulation on the downlink RF signal. Briefly, we leverage the high amplitude period to carry the bus output data while the low amplitude period represents the clock information.

A practical design concern is that the output bus data has a propagation delay because the clock is received from the RF signal. If the delay is close to the period of the high amplitude, the uplink may collapse. Fortunately, the propagation delay is nanoseconds because the clock signal is extracted simply by inverting the PIE envelope (see Fig. 5). In our prototype, we show that the maximum data rate is 200 kbps, which represents a high period of 6.7 μs and it is robust against the delay of 2.7 ns (typically).

Organization of the remainder of this paper. In the following sections, we focus on three important aspects of R2B. First, for the downlink, we describe the technical design of PIE-to-SPI demodulation in § 4. Second, for the uplink, while we have presented the bus signal backscatter methodology, the technical design is described in (§ 5). Third, we further consider multi-chip and multi-device R2B communication in § 6. In the remainder of this paper, we present the implementation, evaluation, discussion, and related work, and we finally give our conclusions.

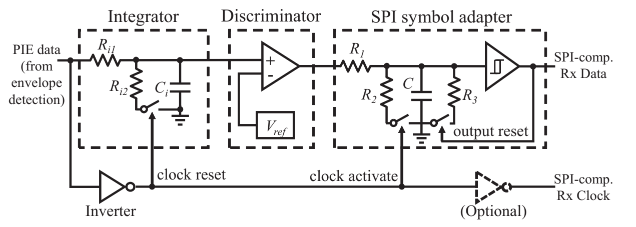


Figure 5: Circuit design for PIE-to-SPI demodulation.

4 PIE-TO-SPI DEMODULATION

In this section, we explain how to convert the PIE baseband to SPI-compatible input data.

Figure 5 illustrates the circuit design for the PIE-to-SPI demodulator. At a high level, the demodulation relies on the fact that the energy contained in PIE "data-0" and "data-1" differs in the logical high period. Thus, an integrator is used to distinguish the PIE data according to the energy difference. Essentially, the integrator incorporates an integration capacitor. When a PIE symbol is input, the logical high voltage is utilized to charge the capacitor. The charged peak voltages are recognizable because "data-0" and "data-1" represent the different durations of the high period. According to this difference, an appropriate threshold exists to discriminate zeros and ones. A robust discrimination threshold should be the middle of the peak voltages between "data-0" and "data-1" in order to mitigate negative effects due to noise. Furthermore, the discriminator output is converted into SPI-compatible bus signals by the SPI symbol adapter. In addition, the clock is extracted by simply inverting the PIE signal. The clock signal is also used to reset the integrator for the next input PIE data. The optional inverter is used to determine the polarity of the clock according to the SPI mode adopted in the SPI bus.

To better understand the SPI symbol adaption design, we first consider the integrator. The clock reset signal controls a single-pole, single-throw (SPST) switch to reset the integrator circuit to its initial state. Specifically, when the integrator is accumulating charge from the PIE data, the switch is open (disconnected) and current flows through R_{i1} into the capacitor C_i. After the clock reset signal arrives, the switch is closed (connected) and the capacitor discharges through R_{i2} to ground. The clock reset aims to prevent integration drift where the next PIE symbol enters before the voltage discharged completely. Thus, R_{i2} should be as small as possible to expedite the discharging process. A similar process occurs in SPI symbol adaption.

The behavior of the SPI symbol adapter is depicted in Fig. 6. SPI "data-0" is simply a logical low voltage that does not need to change, so the problem is the behavior of SPI-compatible "data-1." To address this problem, the symbol adapter must satisfy two requirements. First, the adapter should modify the signal timing to center align the data and clock. Second, the adapter should reshape the data waveform to cover the clock pulse. In our prototype, we adopt the condition of $t_{dat} = 3t_{clk}$, where t_{dat} is the duration of the logical high period for SPI-compatible "data-1" and t_{clk} represents the duration of the clock. To satisfy the requirements stated above,

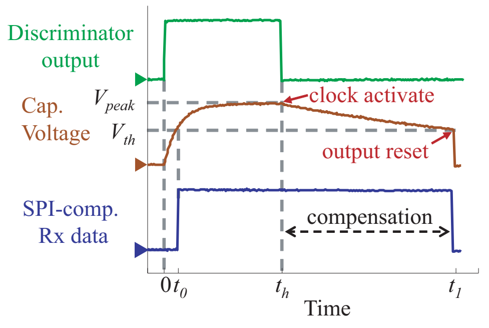


Figure 6: Behavior of the SPI symbol adapter.

the behavior of the adapter is divided into two parts comprising the *delay* and *compensation*.

The *delay* part is given by the charging process, where the logical high voltage (V_h) output from the discriminator charges the capacitor C from zero to V_{peak} through the resistor R_1. Based on the capacitor charging curve, the delayed duration t_{del} is derived by:

$$t_{del} = t_0 = \tau_{del} \ln \frac{V_h}{V_h - V_{th}} + t_{pd} \tag{1}$$

where V_{th} is the threshold of the trigger and t_{pd} is the propagation delay of the data line. Furthermore, $\tau_{del} = R_1 C$ is the RC constant in the *delay* process. By carefully choosing the value of τ_{del}, we determine an appropriate value of t_{del} to shift the data signal so it is center aligned with the clock signal. However, the charging process alone may not provide a logical high period with a sufficient duration to guarantee that $t_{dat} = 3t_{clk}$. Thus, we need additional compensation to extend the logical high period.

The *compensation* part stems from the discharging process driven by two signals. First, compensation starts from the clock signal (similar to the clock reset in the integrator). The capacitor discharges through R_2 to ground. We select a huge resistor value for R_2 to decelerate the decreasing process. Second, when the decreasing voltage is below the trigger threshold V_{th}, the output voltage becomes logically low. We employ a feedback scheme that leverages the falling edge of the output to reset the RC shaper to the initial state. The compensation time t_{com} is provided by the voltage discharging from V_{peak} to V_{th}, as described by:

$$t_{com} = t_1 - t_h = \tau_{com} \ln \frac{V_h(1 - e^{-\frac{t_h}{\tau_{del}}})}{V_{th}} \tag{2}$$

where $\tau_{com} = R_2 C$ represents the RC constant in the compensation part and t_h is the logical high duration of the discriminator output.

Finally, we attempt to find τ_{del} and τ_{com} to realize SPI symbol adaption. The logical high period for SPI "data-1" is described as $t_{dat} = t_1 - t_0$. The data signal should be aligned with the clock, so the clock t_{clk} divides the duration of SPI "data-1" (t_{dat}) into three equal periods in the time domain. Thus, we derive $t_h - t_0 = \frac{1}{3}t_{dat} = t_{clk}$. This observation allows us to substitute the expression for t_0 in Eq. 1 and obtain the following.

$$\tau_{del} = \frac{t_h - t_{clk} - t_{pd}}{\ln \frac{V_h}{V_h - V_{th}}}$$

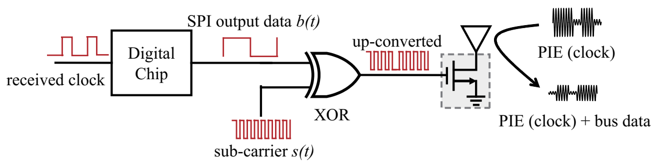


Figure 7: Bus signal backscatter architecture.

In addition, we observe that $t_1 = t_0 + 3t_{clk} = t_h + 2t_{clk}$. We substitute t_1 in Eq. 2 and then calculate the value of τ_{com} as follows.

$$\tau_{com} = \frac{2t_{clk}}{\ln \frac{V_h(1 - e^{-\frac{t_h}{\tau_{del}}})}{V_{th}}}$$

Thus, we have proposed a circuit design for PIE-to-SPI demodulation in this section and provided a model for calculating the key parameters (τ_{del} and τ_{com}) to implement the circuit design. It should be noted that although there are infinite options when selecting the values for the resistor and capacitor, we suggest that the capacitance is as small as possible to reduce the power consumption because the charged voltages in the capacitors will finally be released to ground as energy dissipation.

5 BUS SIGNAL BACKSCATTER

In this section, we present the 2-PSK modulation design for bus signal backscatter on the uplink. In addition, we recommend a receiver design for the gateway to receive the uplink transmissions.

The bus signal backscatter architecture is illustrated in Fig. 7. At a high level, after a received clock enters the chip, the output SPI data are further up-converted by an XOR gate associated with a subcarrier signal. The subcarrier performs a frequency shift at Δf to address the problem of self-interference at the receiver [34, 52]. The XOR gate achieves the 2-PSK modulation where the SPI output data controls the phase change for the subcarrier. Finally, the up-converted signal operates the RF switch to backscatter the downlink transmission signal.

To understand the modulation process, we denote the bus data signal output from the chip as $b(t)$ and the subcarrier signal as $s(t)$, which are both square waves that alternate between +1 and −1 [1]. $s(t)$ is a periodic signal at frequency Δf, so based on Fourier analysis, the subcarrier signal can be written as:

$$s(t) = \frac{4}{\pi} \sum_{n=1,3,5,\cdots}^{\infty} \frac{1}{n} sin(2\pi n \Delta f t)$$

where the first harmonic ($n = 1$) contains the desired sinusoidal signal $\frac{4}{\pi} sin(2\pi \Delta f t)$ and the other harmonics are filtered at the gateway according to the frequency. Furthermore, the up-converted signal output from the XOR gate is described as $\frac{1 - b(t)s(t)}{2}$. The term $b(t)s(t)$ performs the 2-PSK modulation to the subcarrier signal. For instance, when the value of $b(t)$ is −1, the product is treated as a π phase change to the subcarrier, which is denoted as $\frac{4}{\pi} sin(2\pi \Delta f t + \pi)$.

[1]Digital signals are commonly known to toggle between 0 and 1, but the waves finally operate the MOSFET at the antenna to change its impedance. As a result, for the backscatter signal, the square wave is approximated as a sinusoidal signal that alternates between +1 and −1 [15, 19].

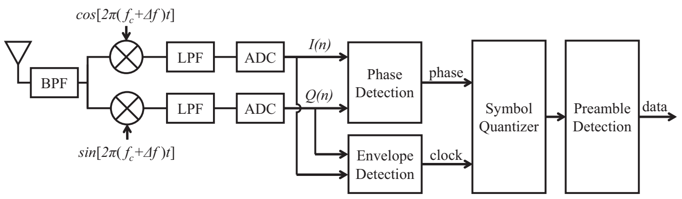


Figure 8: Design of the receiver in a gateway. Recovering bus data from the backscattered signal.

The backscatter operation occurs at the antenna. The downlink transmission signal (ASK modulated with PIE format) from the gateway is written as $PIE(t)cos(2\pi f_c t + \varphi_0)$. As a result, the backscattered signal on the uplink is written as follows.

$$\frac{1}{2} \underbrace{[1 - b(t)s(t)]}_{\text{up-converted signal}} \times \underbrace{PIE(t)cos(2\pi f_c t + \varphi_0)}_{\text{downlink signal}}$$

In the backscattered signal, the desired signal that contains the SPI bus data is written as follows.

$$\frac{2}{\pi} \underbrace{PIE(t)}_{\text{clock}} \underbrace{b(t)}_{\text{data}} \underbrace{sin(2\pi\Delta f t)cos(2\pi f_c t + \varphi_0)}_{\text{carrier}}$$

Receiver Design in the Gateway. The design of the receiver is shown in Fig. 8. The desired backscattered signal is at the center frequency $f_c + \Delta f$, so we first filter the received signal using a band-pass filter centered at $f_c + \Delta f$. Next, we down-convert the signal at two branches to obtain the I and Q components. Furthermore, we extract the baseband signal with a low-pass filter to remove the band noise and an analog-to-digital converter to digitize the signal for subsequent detection. Next, the I and Q components are used to detect the phase and envelope. The phase represents the bus data that are modulated by 2-PSK and the envelope represents the PIE data containing the clock information for symbol synchronization. Furthermore, the phase and clock are input into the symbol quantizer for bit quantization. Finally, we detect the predefined components in the received bit stream and obtain the SPI data from the expected device.

6 R2B MULTI-CHIP NETWORKS

We have explained the design of R2B communication for a single digital chip. In this section, we first consider the multi-chip case where a number of peripheral chips are connected on the SPI bus. We then focus on the case where several R2B devices exist in the vicinity of a gateway. Finally, we discuss the overheads of R2B communication.

Multi-chip R2B Communication. The SPI bus offers every interfaced peripheral a chip select pin. If we activate the pin, the chip becomes active in SPI communication and it is permitted to receive and transmit data on the bus, whereas the others remain on standby. Using this method, multi-chip communication is achieved by selecting each chip in different time slots. In particular, we define a *chip select (CS)* command and design a communication control logic to monitor the received data on the SPI bus. If the data comprises the *CS* command, the control logic will activate the chip described in the command. However, this approach has two problems: (a) the data received for chips may be misunderstood as commands, and (b) the chips would also receive the commands and this will result in errors. To address these problems, we must distinguish the data and commands in multi-chip communication.

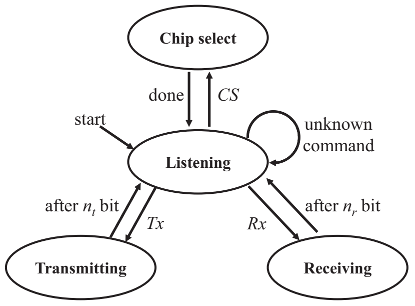


Figure 9: Finite state machine for R2B communication control logic.

The control logic divides the system state into four types using a finite state machine (FSM), as shown in Fig. 9. Initially, the R2B device stays in the listening state. The control logic disables the SPI bus in this state to receive commands. Next, when an *Rx* command arrives, the control logic changes into the receiving state and enables the SPI bus. The receiving state keeps sampling n_r bit data specified in the *Rx* command and returns to the listening state. In addition, a similar transition for the transmitting state occurs in response to a *Tx* command. The amounts of receiving and transmitting data (n_r and n_t) can be set to specific values according to the interfaced sensors. For example, an accelerometer sensor outputs 3×16 bit to indicate three-dimensional information, so we set n_t=48 to allow the gateway to read the collection of data together within one round. In addition, for each transmission, the control logic injects a preamble code into the bus output data before backscattering to allow frame and carrier synchronization at the gateway.

Multi-device R2B Network. Based on the control logic described above, we obtain multi-device networks using an addressing approach. We assign a unique address to each R2B device to be the device's ID stored in a data register. The gateway sends the address before every command to specify the desired R2B device. Only the R2B device with the matching address reacts to the command in the listening state. In addition, when a gateway is communicating with the desired device, other adjacent R2B devices overhear the downlink data but they do nothing apart from listening. However, the downlink data may possibly equal a command for a bystander and lead to an error. To prevent collisions, a longer address length is imposed according to the amount of networked devices. The *reset* command could be used to reboot the device after an error occurs.

R2B Overhead. In R2B communication, the control of the peripheral chips is shifted to the gateway using wireless communication. As a result, the control will occupy the downlink transmission and this incurs communication overheads. In particular, we must consider the following two issues.

(1) Overheads due to chip controls or configurations: We consider a typical timeline when reading data from a sensor, where the

Table 1: RC values for different data rates and bit energy budget for demodulation.

Data Rate	Integrator		RC delayer			Bit Energy Budget
	R_{i1}	C_i	R_1	R_2	C	
30 − 50 kbps	170 KΩ	100 pF	3.2 KΩ	270 KΩ	100 pF	bit 0: 0.47 nJ/bit bit 1: 1.38 nJ/bit
60 − 100 kbps	85 KΩ	100 pF	1.6 KΩ	130 KΩ	100 pF	
90 − 150 kbps	57 KΩ	100 pF	1 KΩ	72 KΩ	100 pF	
120 − 200 kbps	43 KΩ	100 pF	0.8 KΩ	57 KΩ	100 pF	

gateway first sends a command to trigger the sensor and then delivers clocks to read the output data over the air. For output dense chips, frequent commands incur obvious overheads on the downlink. Fortunately, we note that advanced high speed chips (such as accelerometer [4] and memory [2]) support continuous reading that only needs the downlink command once, and the consequent reading operations are performed directly without another command. In addition, some sensors [36] require application-specific configurations and the initialization of many parameters when first used. However, most of the chips store configurations in their nonvolatile memories, thereby avoiding frequent controls during use.

(2) Overheads due to control signaling: Control signaling is used for R2B communication control, including the control logic and device addressing. The control signal is also delivered on the downlink but its proportion is only a small part of the downlink channel usage owing to the simple R2B logic. Thus, compared with the conventional design, R2B incurs higher overheads due to control and signaling to wireless communication, but the overheads are minor in the evaluation. Further details are described in § 8.3.

7 IMPLEMENTATION

In our implementations, we consider two proof-of-concept prototypes. The first prototype is implemented using discrete commercial off-the-shelf (COTS) components to demonstrate the processor-free concept. As discrete components are hard to quantify the power consumption during communication, we then conduct an FPGA implementation to achieve power analysis.

Implementation of Discrete Components. This implementation comprises the PIE-to-SPI demodulator, 2-PSK modulator, and communication control logic (*i.e.*, FSM). In the demodulation process, both the trigger and discriminator use TS3702. The inverter on the clock line is implemented using NL17SZ02DFT2G to achieve a delay on the nanoseconds scale (typically 2.7 ns). The SPST switch employs an NMOS transistor DMG2302UK. Table 1 lists the value for each resistor and capacitor in the demodulator for several data rates. In addition, we set $R_{i2} = R_3 = 0\ \Omega$ to minimize the discharging duration. We show that the prototype achieves a data rate up to 200 kbps. The bit energy budget is related to the capacitance, where a lower capacitance results in a lower energy budget. The minimum capacitance is impacted by the parasitic capacitance of the circuit.

In the modulation process, the subcarrier is generated by LTC6906, which is a micro-power precision oscillator. The XOR gate is implemented by using SN74LV1T86DBVR. In addition, the RF switch BF1212WR coordinates a 2 dBi dipole antenna for backscattering. We also implement the FSM using discrete logic components. Command detection is mainly conducted with the shift register (SN74HC164PW) and an 8-channel NAND gate (CD74HC30PW).

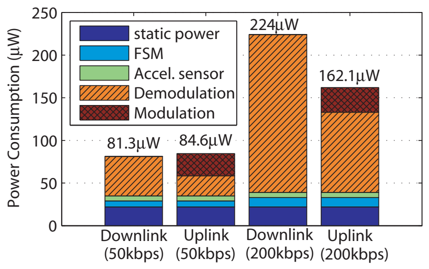


Figure 10: Power consumption breakdown for the FPGA implementation.

The bus control uses a single AND gate (SN74AHC1G08DCK) to enable/disable the bus. The counter mainly comprises a binary counter (SN74LV163ADR). Finally, the flip-flop (NC7SZ175P6X) allows the circuit to keep the system state.

FPGA Implementation and Power Analysis. We implement the FSM for control logic and modulation on a nano-low power FPGA [25] for power analysis. In the modulation process, the SPI output data are used as the inputs for the FPGA. The FPGA generates the sub-carrier at 1 MHz and performs 2-PSK modulation using an XOR gate. Libero SoC [26] is employed to estimate the power consumption. We write Verilog codes and use the default configuration for synthesis, place, and route in the simulation. Furthermore, the power consumption is estimated using the SmartPower tool. The tool shows that the power consumption by the FSM is up to 11 μW at 200 kbps. In addition, the modulation consumes 29.06 μW with a 1 MHz oscillator [37]. Figure 10 plots the power consumption breakdown for the whole R2B device. The modulator only operates when the uplink is used for backscattering, whereas the demodulator is used for both links. The power consumption during demodulation is related to the data rate and the downlink data (bit 0 or 1). On the downlink, we consider the average distributions of bit 0 and bit 1. On the uplink, only the clock is needed, so the downlink data are all bit 0 to reduce the power consumption during demodulation. The power consumption can be reduced further by the ASIC implementation.

8 EVALUATION

Experimental Setup. For comprehensive experiments, we employ a bistatic deployment where two USRPs are configured as a transmitter and a receiver, respectively. We place the R2B device in between the two USRPs in an open area. For convenience, we denote the distance between the transmitter and R2B device as d_1, and the distance between the R2B device and receiver as d_2.

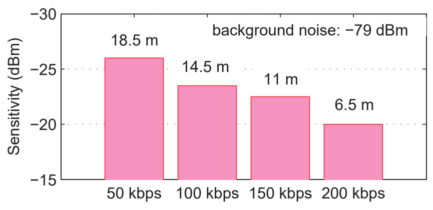


Figure 11: Downlink sensitivity.

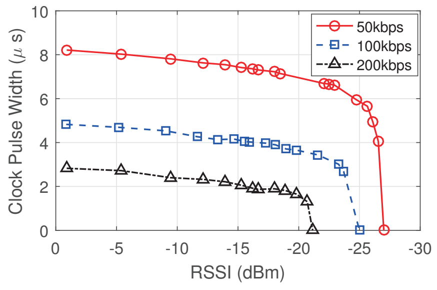


Figure 12: Downlink clock pulse width.

8.1 Bit Error Rate (BER)

BER on the downlink. We employ the transmitter to send predetermined test data to the R2B device. The test data incorporates a group of 8-bit stream "11100010" in a loop 50,000 times. We design the bit stream to cover all bit distributions, including continuous bits and alternate bits for both 0 and 1. For convenience, we connect a micro-controller (MCU) [13] to the R2B device to record the test data. When a byte of data is received, the MCU stores it in the ferroelectric RAM (FRAM) and all of the data are also exported to a PC to subsequently analyze the BER.

We observe that the BER on the downlink is strongly related to the downlink signal power. When the input power is higher than the sensitivity (minimum power of the downlink signal), the communication process achieves a BER of near-zero ($< 10^{-5}$) in our experiments. By contrast, the BER increases dramatically in the vicinity of the sensitivity. Figure 11 plots the measured sensitivities over the data rates and the corresponding maximum downlink range measured in the experiments. We also conduct further experiments and observe that the received SPI clock changes with the distance. Figure 12 shows the clock pulse width as a function of received signal strength (RSSI). As RSSI decreases, the pulse width first declines slowly in a linear manner, before then dropping dramatically to zero and disappearing. The downlink communication BER clearly increases close to the turning points, which we use as the criterion for determining the sensitivity. It is worth to note that the downlink signal power is still much higher than the background noise at the sensitivity. Thus, for passive radio, the signal strength is the most important factor in downlink communication because the passive components demodulate the signal by directly comparing the received signal voltages.

Furthermore, to show the BER when the received signal becomes noisy, we evaluate the downlink BER versus the signal to noise ratio (SNR). As the background noise (around −80 dBm) is much

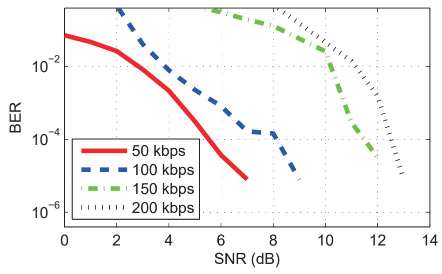


Figure 13: Downlink bit error rate.

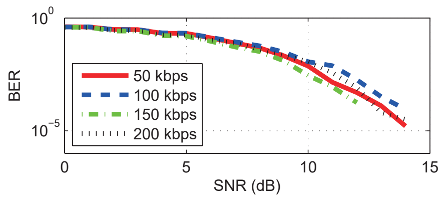


Figure 14: Uplink bit error rate.

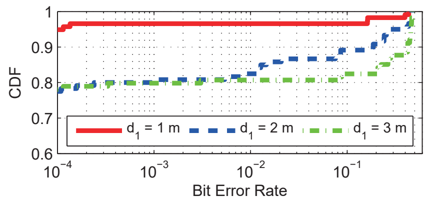


Figure 15: Comprehensive BER when combining the downlink and uplink. d_1 indicates the distance between the transmitter and R2B device.

lower than the signal strength, we use additive white Gaussian noise (AWGN) to increase the noise on the downlink in order to adjust the SNR. The results are shown in Fig. 13. Empirically, the higher data rate indicates the lower energy per bit, which results in greater susceptibility to noise.

BER on the uplink. Similar to the downlink experiment, we operate the USRPs to read predefined data from the R2B device and measure the BER. As shown in Fig. 14, the relationships between BER and SNR are similar to each other at different data rates. Theoretically, the SPI output data are modulated by 2-PSK on the uplink, so the SNR is equal to E_b/N_0 (ratio of energy per bit relative to the noise power spectral density), and thus it is not dependent on the data rate [8]. It should be noted that the SPI symbol is described by both the data and clock, where the clock is provided by the downlink. A higher data rate may lead to a higher probability of an error for the clock signal, but the clock is easy to recover at the receiver due to its periodicity. This observation suggests that if we want to optimize the throughput while reducing the BER, we should select a lower data rate (50 kbps) on the downlink and a higher data rate (200 kbps) on the uplink.

Finally, we determine the comprehensive BER in the overall R2B communication process, as shown in Fig. 15. The experiment is

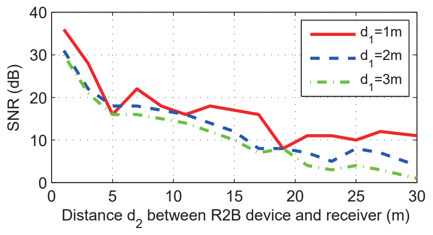


Figure 16: SNR for the uplink signal in the stationary scenario.

conducted based on an echo test where the R2B device first receives data from the downlink and then transmits the same data on the uplink. In each experiment, we fix the transmitter and R2B device at a distance of d_1, and then move the receiver away from the R2B device to the maximum range. The results show that a shorter distance for d_1 can reduce the communication errors in R2B.

8.2 Communication Range

We conduct the communication range experiments in an atrium measuring 18×57 m^2. Our prototype can perform up to a distance of 30 m, so this site is suitable for the evaluations. We consider two deployment scenarios, *i.e.*, stationary and mobile. In the stationary scenario, the R2B device is placed at a fixed point, whereas the mobile scenario considers mobile R2B devices.

Scenario 1 — Stationary R2B device. We place the transmitter and the R2B device at a distance d_1, and then move the receiver away from the R2B device. The measurements are obtained at increments of 1 m. At each distance d_2, we read the memory data from the R2B device. For convenience, the memory data are stored in an SPI register on the MCU. The transmitter sends a *Tx* command with transmission power of 20 dBm at 915 MHz to the R2B device and then delivers clocks to drive the memory data output. The receiver receives the memory data and measures the SNR for the backscatter signal by calculating the ratio of RSSI relative to the background noise. Finally, we repeat these experiments to measure the SNR at several distances d_1. The results are plotted in Fig. 16. As expected, the SNR value decreases along the x-axis. Furthermore, at a certain x-axis value, the increase in d_1 also leads to a reduction in the SNR because the RSSI is negatively related to both distances d_1 and d_2, which is given by the product $d_1 d_2$ [19]. In addition, the measurements are conducted up to 30 m away from the R2B device. At this point, the RSSI in the case where $d_1 = 3m$ is around -80 dBm, which is close to the strength of the background noise.

Scenario 2 — Mobile R2B device. We fix the distance between the transmitter and receiver ($d_1 + d_2$). We then move the R2B device from the transmitter to the receiver at a velocity of 1 m/s. At each second, we measure the RSSI value using the method described above. Figure 17 shows the results obtained at three different distances ($d_1 + d_2$). These results are interesting that if we want to receive a stronger signal in R2B communication, we can move the R2B device close to either the transmitter or the receiver. The lowest SNR occurs near the middle between the two devices. Nevertheless, the effect of the position may be attenuated by increasing the distance ($d_1 + d_2$) between the transmitter and the receiver.

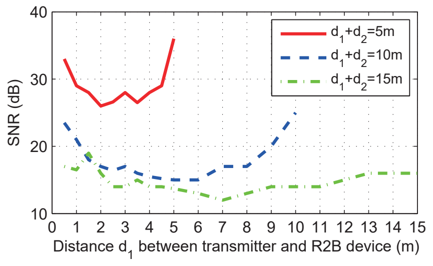


Figure 17: SNR for the uplink signal in the mobile scenario. We fix the distance ($d_1 + d_2$) between the transmitter and the receiver, and move the R2B device from the transmitter to the receiver at a speed of 1 m/s.

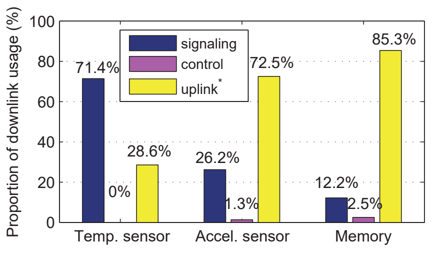


Figure 18: Downlink usage when reading data from various chips. *Uplink needs downlink to deliver clock.

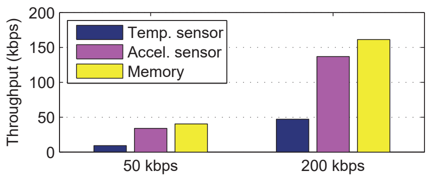


Figure 19: End-to-end throughput.

8.3 Channel Usage and R2B Throughput

Given that the transmissions for both uplink and control signaling need to utilize part of the downlink, we evaluate the downlink channel usage. Further, we evaluate the end-to-end throughput for a peripheral chip talking to the gateway via R2B communication.

We consider three peripheral chips, *i.e.*, a temperature sensor [14], accelerometer [4], and FRAM memory [2]. For each chip, we operate the USRPs to read 1 KB (sensed or stored) data from the R2B device, and measure the downlink usage. Figure 18 plots the proportion of downlink channel usage. The temperature sensor does not support the configuration, and thus it accounts for 0% of the downlink control usage. Moreover, every temperature conversion has to be launched by controlling the chip select pin. Thus, signaling is used to achieve control, which results in major downlink usage. In addition, the accelerometer and memory support continuous reading, so they account for a small proportion of the channel usage for downlink controls. Finally, the uplink needs the downlink to provide clocks, so a larger proportion of the uplink usage leads to higher throughput for reading the chip data.

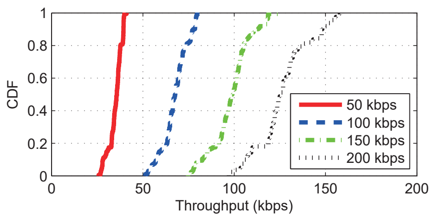


Figure 20: Network throughput over data rates.

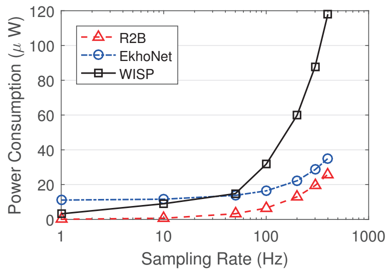


Figure 21: Power consumption comparison.

Figure 19 shows the throughput for each peripheral chip over the data rate. As expected, a higher data rate can efficiently improve the throughput. Moreover, for the continuous reading supported chips, R2B communication typically obtains a higher throughput compared with the non-continuous reading chips (such as the temperature sensor). These results suggest interesting design choices for R2B devices, where sensors with continuous reading functions should be the first option if we desire high throughput in practical applications.

8.4 Network Throughput

In the evaluation of the network throughput, we consider a simple R2B network containing four R2B devices. Each device is installed with a peripheral chip, *i.e.*, temperature sensor [14], accelerometer [4], micro-controller [13], or memory [2]. In every time slot, the USRP randomly selects an R2B device via the device address and also accesses the peripheral chip (read sensors or memories). The average payload for each access is 184 bits for each device. The receiver USRP is configured to record the number of data on both the downlink and uplink.

The throughput measured over different data rates is shown in Fig. 20. The data rate is the bottleneck for the network throughput. For each data rate, the practical network throughput is mainly affected by the response delay. Specifically, the transmitter USRP requires time delays to wait for the response from the peripheral, such as completion of the temperature conversion process. Thus, throughput optimization is required for the R2B network, where if the duration of the response delay is sufficient to access another chip, the USRP can talk to other R2B devices to efficiently use the channel. However, if the duration is excessively short to communicate with others, the USRP must wait for the response, which will reduce the throughput. Finally, we determine the performance of the network. The experiment demonstrates the feasibility of R2B networks, but it would be challenging to support large numbers of devices in the network, such as hundreds or thousands of devices. Indeed, the current network architecture only has one wireless channel available to operate all of the sensors and deliver control signaling. Increasing the number of chips in the network may lead to a low response rate. Technologies such as CDMA and FDMA may improve the response rate to allow multiple access.

8.5 Power Benefits

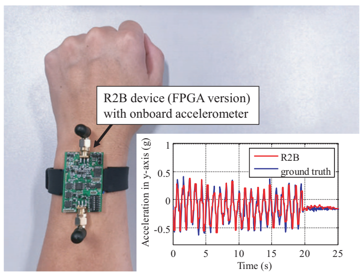
We acknowledge that it is quite difficult to find suitable power consumption baselines to compare with R2B. R2B achieves a direct connection between the backscatter radio and SPI sensors, thereby shifting the SPI master to the remote gateway via wireless. Given this, we consider EkhoNet [50] and WISP5.0 as the baselines, where the former comprises an SPI master implemented using hardware (*e.g.*, FPGA) and the latter employs the software SPI master using an MCU in the end device. We conduct experiments to interact with an SPI accelerometer (ADXL362) and measure the power consumption for comparison. The experimental setup is the same for EkhoNet, where R2B and EkhoNet are implemented on the same Igloo Nano FPGA and the crystal oscillator. In addition, to ensure a fair comparison, the MCU on the WISP includes a deep sleep schedule to save energy whenever possible.

Figure 21 shows the power consumption results. R2B consumes 25.9 μW at a sampling rate of 400 Hz, which is 4.5× lower than the 118 μW consumed by the WISP as the conventional design. In addition, at 400 Hz, R2B reduces the power consumption by 26% compared with EkhoNet by eliminating the overheads due to the local SPI master. Furthermore, R2B has a lower static current consumption than EkhoNet because of its simpler architecture. In particular, the EkhoNet implementation involves 6K gates, whereas our FPGA implementation has 57 look-up tables and 48 flip-flops, involving a total of 737 gates. In addition to the power benefits, R2B also eliminates the programming demand on the sensor platform due to the wireless SPI control. In our gateway implementation, the USRP 2922 runs LabVIEW to remotely operate the sensors in the end device. The logic in the end device is kept simple and it requires no adaptation to interface different SPI sensors.

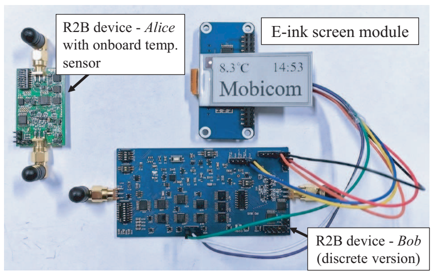
8.6 Proof-of-Concept Applications

Programming-free. We build a programming-free wristband with R2B as an application based on acceleration measurements, as shown in Fig. 22(a). The on-board accelerometer is directly operated by the USRP via the wireless SPI bus over the air. We compare the results with the ground truth, where the conventional design utilizes an MSP430 MCU [12] to control and read the sensor.

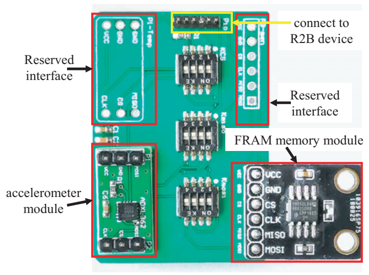
Chip-scale Network. To illustrate the Internet of microchips concept, we test an example comprising a wireless indoor temperature display. As shown in Fig. 22(b), the experiment includes two R2B devices, where *Alice* is equipped with a temperature sensor [14] and *Bob* with an e-ink screen [5]. The gateway first reads the temperature from *Alice* and then plots the measurement results on *Bob*'s screen, although they do not know each other. The network is fully flexible and scalable. We could easily add other R2B devices


R2B device (FPGA version)
with onboard accelerometer
R2B
ground truth
Acceleration in y-axis (g)
Time (s)
1
0.5
0
-0.5
0
5
10
15
20
25

(a) Programming-free wristband.



(b) Wireless temperature display.



(c) Plug and play interface.

Figure 22: Proof-of-concept applications based on R2B communication.

into the network such as equipping *Carol* with a sensor for humidity measurement. These configurations only require updating the software in the gateway.

Plug and Play. The bus topology allows us to easily install new peripheral modules to extend the functions for R2B devices. As shown in Fig. 22(c), we build a simple plug-and-play interface to improve the convenience of our experiments. Exploring more advanced schemes (*e.g.*, fully automatic) for plug-and-play deployment would be an interesting future direction for IoT scenarios.

Rapid software development. In the applications described above, we observe that it is easier and faster to develop software for an R2B application rather than the conventional IoT devices. Traditionally, it is necessary to write embedded C or assembly language codes to operate registers, IOs and interrupts, and to deal with hardware contexts. In the processor-free architecture, embedded software can be replaced with Matlab codes in the gateway instead of sending the bit stream and analyze the received bus data. Other high-level languages will be supported in the future, *e.g.*, Python and Java.

9 DISCUSSION AND LIMITATIONS

In this paper, we provide a proof-of-concept for the Internet-of-Microchips as an early step toward this vision. There are some design concerns and limitations in the current prototype, as discussed in the following.

SPI Clock Timing. The standard SPI clock is frequency fixed with a constant duty cycle, but the clock signal in our design has a variable frequency over different downlink data due to the PIE format. Can the SPI still work correctly on a PIE-providing clock? To answer this question, it is necessary to acknowledge that SPI involves synchronous serial communication, where the data transfer is fully driven by the clock signal. Communication correction depends on the synchronization between the clock and data. In our design, the PIE format embeds clock information within each bit, which has satisfied the synchronization requirements on the downlink. Furthermore, PIE-to-SPI demodulation retains the synchronization with the SPI. We conduct experiments to verify the successful communication up to 200 kbps.

Data Rate. The current prototype only achieves a data rate up to 200 kbps, but theoretically the SPI bus can support a much higher data rate (often over 1 Mbps). A key technical hurdle to improve the data rate is the downlink coding and demodulation scheme, where the length of PIE symbols decreases along with data rates, indicating more challenges to discriminate the PIE data by the analog circuit. Thus, the current prototype can only interface low rate sensors. Future studies can investigate the development of a more efficient coding scheme than PIE and an anti-noise demodulation circuit in order to obtain a higher data rate.

Event-triggered Sensing. The current design does not support event-triggered sensing. Some sensors have a trigger pin to interrupt and inform the master (such as MCU) regarding the detection of events or the completion of sensing tasks. This type of functionality is valuable because it avoids frequent queries from the master. An alternative solution in our design involves the gateway setting a time delay before reading the uplink data. The delay is obtained based on the sensor specifications. A more comprehensive design can be attempted in future research, where the successful development of this design would be highly beneficial for the proposed system but also for a broad range of backscatter platforms.

Security and Privacy. This paper does not consider security in terms of data privacy, where wireless transmissions may be overheard by adjacent devices. This problem could be overcome by using an addressing approach. In particular, we have described the use of a unique address for each R2B device for networking in § 6. We can leverage the address to encrypt the communication to allow security and privacy enhancement. Encryption (decryption) in R2B devices can be achieved by using an XOR gate in the modulation (demodulation) process. Thus, only the desired R2B device can decode the commands and data on the downlink, and only the receiver who knows the address can receive the correct bus data on the uplink.

Error Detection and Re-transmission. We have presented an evaluation based on the BER in § 8.1, but our current prototype does not consider data error detection and the re-transmission mechanism. To address this, we can implement a data checksum function in the FSM to detect bit errors in the received data. If bit errors occur, the R2B device can request re-transmissions from the gateway by controlling the bus to backscatter a specific error code. The analogous operation can be implemented in the FSM to insert a checksum behind each data transmission on the uplink.

Processor-free Computation. Based on the R2B paradigm, we propose a processor-free architecture for IoT end devices. It should be noted that our aim is not to fully eliminate computations, but to reduce redundancies in the devices in order to lower the power consumption, cost, and complexity. Thus, an application-specific processor is still optional in certain cases when necessary local computations must be performed (*e.g.*, in-sensor calibration). In these cases, we can treat the processor as one of the peripherals or it may be embedded in the control logic. Large amounts of computations are shifted to the gateway and the processor's capability can be reduced according to application-specific requirements in order to obtain power and cost savings.

10 RELATED WORK

Our proposed method is related to RF backscatter communication. According to the baseband source, the existing backscatter techniques are classified as device-scale and chip-scale categories. Device-scale backscatter means that the baseband signal in the backscatter device is provided by a digital core (*e.g.*, a microprocessor). The digital core generates (receives) a baseband signal representing the information transmitted to (received from) another device. Alternatively, chip-scale backscatter indicates that the baseband is the input/output of a chip (*e.g.*, a sensor). The related research is summarized in the following.

Device-scale RF Backscatter. Several methods have been proposed that employ ambient signals to perform backscatter communication. Ambient backscatter reflects TV signals in the vicinity of TV towers [23, 29]. Wi-Fi backscatter [18] creates CSI changes in existing Wi-Fi packets by reflecting the transmissions. FM backscatter [45] utilizes the ubiquitous nature of FM signals. Additionally, TunnelScatter [43] enables transmissions when there is no ambient signal by using a tunnel diode-based RF oscillator. Furthermore, recent studies have demonstrated the ability to generate Wi-Fi [15, 19], BLE [6], Zigbee [21, 31], and Lora [22, 28, 30, 39] signals via backscatter to enable COTS devices (*e.g.*, smartphones) as the signal receiver. In addition, BackFi [1], Buzz [46], LF backscatter [10], in-body backscatter [44], and other systems [9, 24, 47–49, 51–53] can all be classified as device-scale RF backscatter communication.

While device-scale backscatter radios have a typical μW budget for wireless communication, at the system level, a microprocessor is indispensable for operating sensors and processing data. Thus, the IoT end devices with device-scale backscatter have the conventional architecture. In order to reduce the power consumption under high throughput conditions, EkhoNet [50] addresses the power consumption problem for microprocessors in backscatter devices and optimizes the power consumption by shifting the computational overheads from the end device to the gateway. Compared with EkhoNet, our basic design further shifts the sensing subsystem (*i.e.*, sensor control) to the gateway and directly connected sensors to the backscatter communication subsystem via the SPI bus, which obtains power benefits and also eliminates the programming demand on the backscatter device.

Chip-scale RF Backscatter. The early chip-scale research can be traced back to the battery-free cellphone paper [40], where the output from an electret microphone operates the antenna to backscatter the voice of the speaker. Subsequently, methods are proposed for connecting analog sensors to a reader. Battery-free video streaming [27] involves analog video backscatter to modulate the output voltages from an image sensor for backscattering. Varshney *et al*. [42] connect a visible light sensor directly to a reader using backscatter for hand gesture sensing. RF Bandaid [32] provides a fully-analog backscatter platform for interfacing analog sensors. Moreover, hybrid A/D backscatter [41] employs a processor to allow digital controllability for analog sensors.

The methods described above shift digitization and computation to a powerful reader, and they achieve ultra-low power consumption in devices while even outputting at high rates. However, analog chips are more vulnerable to noise and small variations in the voltage, which could lead to errors. Digital chips are more reliable for future applications in sensors because of their noise tolerance, accuracy with field calibration [38], and reliability due to on-chip self-diagnostics [20]. Furthermore, digital chips can provide devices with control, memory, and intelligence (computing). Increasingly, low-power sensors are going to cover most of the demands for IoT applications, such as accelerometers, cameras, and microphones. Our proposed R2B communication allows digital chips to be employed without processors. We consider that the R2B concept is suitable for passive radios via backscatter, but also for rethinking the design of active radios or their combinations [11, 33].

Design References in R2B. The design of R2B communication is based on some previous studies of RFID systems. The design of PIE symbols for data and clock transmission is inspired by the EPC C1G2 protocol [7]. Differently, we exploit the PIE symbol on both the downlink and uplink for bus data output and backscatter transmission. Moreover, we duplicate the RF envelope detection circuit in [35] but the internal five-stage voltage multiplier is changed into a two-stage version to optimize the communication distance. In addition, the paper [17] proposes a PIE decoder design to discriminate the PIE data for RFID tag IC, but the output is only used by the internal FSM and it cannot communicate with the SPI bus.

11 CONCLUSION AND FUTURE WORK

In this paper, we propose a processor-free design for IoT end devices. In particular, we propose R2B communication to allow the backscatter radio to directly control the peripherals. We design the PIE-to-SPI demodulation process to enable downlink transmission. The proposed method employs the bus signal backscatter to achieve uplink transmission. We also explain the control logic for organizing a simple R2B network. Finally, we evaluate our design and show the potential benefits of the Internet-of-Microchips architecture, including a programming-free sensor platform, plug-and-play deployment, and rapid system development.

Future research may focus on three directions. First, future studies may aim to improve the R2B communication performance, including the data rate, network throughput, and concurrency. Second, for gateways, it is compelling to enable COTS mobile devices (*e.g.*, smartphones) to communicate with R2B devices using existing wireless protocols. Third, for the applications described in § 8.6, future studies may involve deeper investigations to make the applications more reliable. In addition, more Internet-of-Microchips applications could be developed to improve IoT smart devices.

ACKNOWLEDGMENTS

We sincerely thank our anonymous shepherd and all reviewers for their time and efforts in reviewing our work and providing valuable feedback. This work was supported in part by the National Natural Science Foundation of China under Grant 61872061, and in part by the Singapore MOE Tier 1 Grant No. RG129/18.

REFERENCES

[1] Dinesh Bharadia, Kiran Raj Joshi, Manikanta Kotaru, and Sachin Katti. 2015. BackFi: High Throughput WiFi Backscatter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. ACM, New York, NY, USA, 283–296. https://doi.org/10.1145/2785956.2787490

[2] Cypress Semiconductor Corporation. 2018. 64-Kbit SPI F-RAM. https://www.cypress.com/file/41676/download.

[3] Jari-Pascal Curty, Michel Declercq, Catherine Dehollain, and Norbert Joehl. 2006. *Design and optimization of passive UHF RFID systems*. Springer Science & Business Media.

[4] Analog Devices. 2019. Analog devices adxl362 accelerometer sensor. https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf.

[5] Waveshare Electronics. 2019. 212x104, 2.13inch E-Ink. https://www.waveshare.com/wiki/2.13inch_e-Paper_HAT_(B).

[6] Joshua F Ensworth and Matthew S Reynolds. 2015. Every smart phone is a backscatter reader: Modulated backscatter compatibility with Bluetooth 4.0 Low Energy (BLE) devices. In *2015 IEEE International Conference on RFID (RFID)*. 78–85. https://doi.org/10.1109/RFID.2015.7113076

[7] EPCglobal. 2008. EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 960 MHz Version 1.2.0.

[8] Andrea Goldsmith. 2005. *Wireless communications*. Cambridge university press.

[9] Mehrdad Hessar, Ali Najafi, and Shyamnath Gollakota. 2019. Netscatter: Enabling Large-Scale Backscatter Networks. In *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation* (Boston, MA, USA) *(NSDI'19)*. USENIX Association, USA, 271–283.

[10] Pan Hu, Pengyu Zhang, and Deepak Ganesan. 2015. Laissez-Faire: Fully Asymmetric Backscatter Communication. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. ACM, New York, NY, USA, 255–267. https://doi.org/10.1145/2785956.2787477

[11] Pan Hu, Pengyu Zhang, Mohammad Rostami, and Deepak Ganesan. 2016. Braidio: An Integrated Active-Passive Radio for Mobile Devices with Asymmetric Energy Budgets. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM'16)*. Association for Computing Machinery, New York, NY, USA, 384–397. https://doi.org/10.1145/2934872.2934902

[12] Texas Instruments. 2012. MSP430F2132 Datasheet. http://www.ti.com.cn/cn/lit/ds/symlink/msp430f2132.pdf.

[13] Texas Instruments. 2018. MSP430FR5969 datasheet. http://www.ti.com/lit/ds/symlink/msp430fr5969.pdf.

[14] Maxim Integrated. 2016. 12-Bit Digital Temperature Sensors with Serial Interface. https://datasheets.maximintegrated.com/en/ds/MAX6629-MAX6632.pdf.

[15] Vikram Iyer, Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua Smith. 2016. Inter-Technology Backscatter: Towards Internet Connectivity for Implanted Devices. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM'16)*. Association for Computing Machinery, New York, NY, USA, 356–369. https://doi.org/10.1145/2934872.2934894

[16] Hrishikesh Jayakumar, Kangwoo Lee, Woo Suk Lee, Arnab Raha, Younghyun Kim, and Vijay Raghunathan. 2014. Powering the Internet of Things. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design* (La Jolla, California, USA) *(ISLPED '14)*. Association for Computing Machinery, New York, NY, USA, 375–380. https://doi.org/10.1145/2627369.2631644

[17] Udo Karthaus and Martin Fischer. 2003. Fully integrated passive UHF RFID transponder IC with 16.7-/spl mu/W minimum RF input power. *IEEE Journal of solid-state circuits* 38, 10 (2003), 1602–1608.

[18] Bryce Kellogg, Aaron Parks, Shyamnath Gollakota, Joshua R. Smith, and David Wetherall. 2014. Wi-Fi Backscatter: Internet Connectivity for RF-Powered Devices. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 607–618. https://doi.org/10.1145/2619239.2626319

[19] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R. Smith. 2016. Passive Wi-Fi: Bringing Low Power to Wi-Fi Transmissions. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation* (Santa Clara, CA) *(NSDI'16)*. USENIX Association, USA, 151–164.

[20] Oscar J Kuiken, Xiao Zhang, and Hans G Kerkhoff. 2008. Built-in self-diagnostics for a NoC-based reconfigurable IC for dependable beamforming applications. In *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*. IEEE, 45–53.

[21] Yan Li, Zicheng Chi, Xin Liu, and Ting Zhu. 2018. Passive-ZigBee: Enabling ZigBee Communication in IoT Networks with 1000X+ Less Power Consumption. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems* (Shenzhen, China) *(SenSys '18)*. Association for Computing Machinery, New York, NY, USA, 159–171. https://doi.org/10.1145/3274783.3274846

[22] Jansen C Liando, Amalinda Gamage, Agustinus W Tengourtius, and Mo Li. 2019. Known and unknown facts of LoRa: Experiences from a large-scale measurement study. *ACM Transactions on Sensor Networks (TOSN)* 15, 2 (2019), 1–35.

[23] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R. Smith. 2013. Ambient Backscatter: Wireless Communication out of Thin Air. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (Hong Kong, China) *(SIGCOMM '13)*. Association for Computing Machinery, New York, NY, USA, 39–50. https://doi.org/10.1145/2486001.2486015

[24] Vincent Liu, Vamsi Talla, and Shyamnath Gollakota. 2014. Enabling Instantaneous Feedback with Full-duplex Backscatter. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking* (Maui, Hawaii, USA) *(MobiCom '14)*. ACM, New York, NY, USA, 67–78. https://doi.org/10.1145/2639108.2639136

[25] Microsemi. 2015. IGLOO nano Low Power Flash FPGAs. https://www.microsemi.com/document-portal/doc_view/130695-ds0110-igloo-nano-low-power-flash-fpgas-datasheet.

[26] Microsemi. 2019. Libero SoC. https://www.microsemi.com/product-directory/design-resources/1750-libero-soc.

[27] Saman Naderiparizi, Mehrdad Hessar, Vamsi Talla, Shyamnath Gollakota, and Joshua R. Smith. 2018. Towards Battery-Free HD Video Streaming. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation* (Renton, WA, USA) *(NSDI'18)*. USENIX Association, USA, 233–247.

[28] Rajalakshmi Nandakumar, Vikram Iyer, and Shyamnath Gollakota. 2018. 3D Localization for Sub-Centimeter Sized Devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems* (Shenzhen, China) *(SenSys '18)*. Association for Computing Machinery, New York, NY, USA, 108–119. https://doi.org/10.1145/3274783.3274851

[29] Aaron N. Parks, Angli Liu, Shyamnath Gollakota, and Joshua R. Smith. 2014. Turbocharging Ambient Backscatter Communication. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 619–630. https://doi.org/10.1145/2619239.2626312

[30] Yao Peng, Longfei Shangguan, Yue Hu, Yujie Qian, Xianshang Lin, Xiaojiang Chen, Dingyi Fang, and Kyle Jamieson. 2018. PLoRa: A Passive Long-Range Data Network from Ambient LoRa Transmissions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 147–160. https://doi.org/10.1145/3230543.3230567

[31] Carlos Pérez-Penichet, Frederik Hermans, Ambuj Varshney, and Thiemo Voigt. 2016. Augmenting IoT networks with backscatter-enabled passive sensor tags. In *Proceedings of the 3rd Workshop on Hot Topics in Wireless*. ACM, 23–27.

[32] Vaishnavi Ranganathan, Sidhant Gupta, Jonathan Lester, Joshua R. Smith, and Desney Tan. 2018. RF Bandaid: A Fully-Analog and Passive Wireless Interface for Wearable Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 2, Article 79 (July 2018), 21 pages. https://doi.org/10.1145/3214282

[33] Mohammad Rostami, Jeremy Gummeson, Ali Kiaghadi, and Deepak Ganesan. 2018. Polymorphic Radios: A New Design Paradigm for Ultra-Low Power Communication. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 446–460. https://doi.org/10.1145/3230543.3230571

[34] Mohammad Rostami, Karthik Sundaresan, Eugene Chai, Sampath Rangarajan, and Deepak Ganesan. 2020. Redefining Passive in Backscattering with Commodity Devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom'20)*. Association for Computing Machinery, New York, NY, USA, Article 3, 13 pages. https://doi.org/10.1145/3372224.3380880

[35] Alanson P Sample, Daniel J Yeager, Pauline S Powledge, Alexander V Mamishev, and Joshua R Smith. 2008. Design of an RFID-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement* 57, 11 (2008), 2608–2615.

[36] Bosch Sensortec. 2018. Humidity and pressure sensor. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf.

[37] SiTime. 2018. SiT1576 μPower, Low-Jitter 1Hz - 2.5 MHz Oscillator. https://www.sitime.com/datasheet/SiT1576.

[38] Laurent Spinelle, Michel Gerboles, Maria Gabriella Villani, Manuel Aleixandre, and Fausto Bonavitacola. 2015. Field calibration of a cluster of low-cost available sensors for air quality monitoring. Part A: Ozone and nitrogen dioxide. *Sensors and Actuators B: Chemical* 215 (2015), 249 – 257. https://doi.org/10.1016/j.snb.2015.03.031

[39] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R. Smith, and Shyamnath Gollakota. 2017. LoRa Backscatter: Enabling The Vision of Ubiquitous Connectivity. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 105 (Sept. 2017), 24 pages. https://doi.org/10.1145/3130970

[40] Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua R. Smith. 2017. Battery-Free Cellphone. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 25 (June 2017), 20 pages. https://doi.org/10.1145/3090090

[41] V. Talla and J. R. Smith. 2013. Hybrid analog-digital backscatter: A new approach for battery-free sensing. In *2013 IEEE International Conference on RFID (RFID)*. 74–81. https://doi.org/10.1109/RFID.2013.6548138

[42] Ambuj Varshney, Andreas Soleiman, Luca Mottola, and Thiemo Voigt. 2017. Battery-free visible light sensing. In *Proceedings of the 4th ACM Workshop on*

Visible Light Communication Systems. ACM, 3–8.

[43] Ambuj Varshney, Andreas Soleiman, and Thiemo Voigt. 2019. TunnelScatter: Low Power Communication for Sensor Tags Using Tunnel Diodes. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 50, 17 pages. https://doi.org/10.1145/3300061.3345451

[44] Deepak Vasisht, Guo Zhang, Omid Abari, Hsiao-Ming Lu, Jacob Flanz, and Dina Katabi. 2018. In-body Backscatter Communication and Localization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. ACM, New York, NY, USA, 132–146. https://doi.org/10.1145/3230543.3230565

[45] Anran Wang, Vikram Iyer, Vamsi Talla, Joshua R. Smith, and Shyamnath Gollakota. 2017. FM Backscatter: Enabling Connected Cities and Smart Fabrics. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation* (Boston, MA, USA) *(NSDI'17)*. USENIX Association, USA, 243–258.

[46] Jue Wang, Haitham Hassanieh, Dina Katabi, and Piotr Indyk. 2012. Efficient and Reliable Low-Power Backscatter Networks. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (Helsinki, Finland) *(SIGCOMM '12)*. Association for Computing Machinery, New York, NY, USA, 61–72. https://doi.org/10.1145/2342356.2342364

[47] Pengyu Zhang, Dinesh Bharadia, Kiran Joshi, and Sachin Katti. 2016. HitchHike: Practical Backscatter Using Commodity WiFi. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM* (Stanford, CA, USA) *(SenSys '16)*. Association for Computing Machinery, New York, NY, USA, 259–271. https://doi.org/10.1145/2994551.2994565

[48] Pengyu Zhang and Deepak Ganesan. 2014. Enabling Bit-by-Bit Backscatter Communication in Severe Energy Harvesting Environments. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation* (Seattle, WA) *(NSDI'14)*. USENIX Association, USA, 345–357.

[49] Pengyu Zhang, Jeremy Gummeson, and Deepak Ganesan. 2012. BLINK: A High Throughput Link Layer for Backscatter Communication. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (Low Wood Bay, Lake District, UK) *(MobiSys '12)*. ACM, New York, NY, USA, 99–112. https://doi.org/10.1145/2307636.2307646

[50] Pengyu Zhang, Pan Hu, Vijay Pasikanti, and Deepak Ganesan. 2014. Ekhonet: High speed ultra low-power backscatter for next generation sensors. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 557–568.

[51] Pengyu Zhang, Colleen Josephson, Dinesh Bharadia, and Sachin Katti. 2017. FreeRider: Backscatter Communication Using Commodity Radios. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies* (Incheon, Republic of Korea) *(CoNEXT '17)*. ACM, New York, NY, USA, 389–401. https://doi.org/10.1145/3143361.3143374

[52] Pengyu Zhang, Mohammad Rostami, Pan Hu, and Deepak Ganesan. 2016. Enabling Practical Backscatter Communication for On-Body Sensors. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 370–383. https://doi.org/10.1145/2934872.2934901

[53] Renjie Zhao, Fengyuan Zhu, Yuda Feng, Siyuan Peng, Xiaohua Tian, Hui Yu, and Xinbing Wang. 2019. OFDMA-Enabled Wi-Fi Backscatter. In *The 25th Annual International Conference on Mobile Computing and Networking* (Los Cabos, Mexico) *(MobiCom '19)*. ACM, New York, NY, USA, Article 20, 15 pages. https://doi.org/10.1145/3300061.3300121