Everything has its Bad Side and Good Side: Turning Processors to Low Overhead Radios Using Side-Channels

Justin Feng
jfeng10@ucla.edu
UCLA
Los Angeles, California, USA

Timothy Jacques
tjacques888@ucla.edu
UCLA
Los Angeles, California, USA

Omid Abari
omid@cs.ucla.edu
UCLA
Los Angeles, California, USA

Nader Sehatbakhsh
nsehat@ee.ucla.edu
UCLA
Los Angeles, California, USA

ABSTRACT

Side-channels have traditionally been exploited as a means of uncovering sensitive information such as cryptographic keys from a computing device. In particular, past work has shown that electromagnetic (EM) radiation from a device's processor and memory during the execution of code and data can be used by attackers to extract private information. In contrast, instead of considering side-channels and electromagnetic radiation as vulnerabilities, we see them as opportunities for wireless communication on resource-limited IoT devices. We present SideComm, a side-channel-based communication system that leverages processors' EM side-channels to enable resource-limited IoT devices to wirelessly send their data without having any radios. The main advantage of this approach is completely eliminating the need for a conventional radio and antenna, which offers energy savings, simplicity, and flexibility for IoT devices. Our evaluation demonstrates SideComm's ability to achieve a communication range of more than 10m (enabling ≥3 dB SNR at 15m) and to work in non-line-of-sight scenarios, such as around corners and through walls. We believe SideComm can enable increased connectivity for many resource-constrained IoT devices in smart environments.

CCS CONCEPTS

• **Computer systems organization** → **Embedded software**; **Embedded hardware**; • **Networks** → *Cross-layer protocols.*

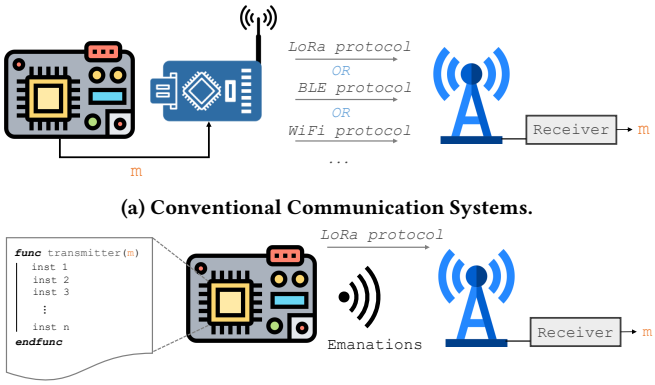
KEYWORDS

Physical side-channels, embedded systems, low overhead communication

ACM Reference Format:
Justin Feng, Timothy Jacques, Omid Abari, and Nader Sehatbakhsh. 2023. Everything has its Bad Side and Good Side: Turning Processors to Low Overhead Radios Using Side-Channels. In *The 22nd International Conference on Information Processing in Sensor Networks (IPSN '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3583120.3586959

IPSN '23, May 09–12, 2023, San Antonio, TX, USA

ACM ISBN 979-8-4007-0118-4/23/05.
https://doi.org/10.1145/3583120.3586959



(a) Conventional Communication Systems.

(b) Proposed Side-Channel-Based Communication System.

Figure 1: Instead of using a conventional physical antenna and a transmitter, SideComm uses software-controlled unintentional RF radiation from the circuit to create controlled RF packets and does not need an antenna.

1 INTRODUCTION

Often an unwanted artifact of any computation is an unintentional digital and/or analog signal created by the computing device. These signals, referred to as side-channel signals, convey information about the code and data being executed on the hardware. An important class of analog side-channel signals is electromagnetic (EM) emanations, which create unintentional RF radiation from the processor's board at various frequency ranges and can be received from some distance using conventional RF receivers.

Traditionally, side-channels are largely considered a vulnerability and have been exploited to infer sensitive attributes of the device, such as cryptographic keys [9, 40, 57]. However, the fundamental relationship between the actual instructions being executed in software and the created side-channels suggests that these unwanted RF radiations could be potentially used for useful purposes.

In this paper, we propose using EM side-channels for ***useful communication***. Our key idea is to leverage the unintentional RF emissions created by the switching activities of a microprocessor in

order to create a specific communication modulation and protocol and hence enabling data transmission. These unintentional RF radiations (or as we call them in this paper, EM side-channels), are an artifact of computing different instructions — i.e., executing various instructions on the microprocessor results in creating a different RF pattern (both amplitude and frequency) which can be observed and collected at some distance from the device. This approach is different from existing work that uses DRAM memory accesses to create the modulation [8, 41], an approach that is *only* applicable to resourceful devices, such as PCs, servers, and mobile computing systems, and not low-end IoT devices. Our work, instead, focuses on *low-power IoT devices* and generates and modulates the desired RF signal (e.g., based on a specific communication protocol) by carefully crafting a series of software activities. This novel method to transmit data has multiple advantages:

(1) **Simplicity and Low-Overhead.** Our approach is very simple and low-cost. In particular, by turning a processor into a radio, we eliminate the need for antennas, RF radios, and baseband processors. Most IoT sensors are already equipped with a processor (or a microcontroller), and our solution reuses it to transmit data directly from it to the air.
(2) **No Need for DRAM and Applicable to IoTs.** Unlike state-of-the-art [8, 41] which only works with devices with DRAM, our method works on resource-constrained bare-metal IoTs (e.g., Arduino, MSP430, etc.).
(3) **Flexibility**. Our RF signal is directly generated by a processor and purely controlled by software. Therefore, changing the configurations to support different rates or protocols (beyond the modulation shown in this paper) is as simple as changing a function in a program to match the requirements. This is in contrast to conventional radios, which require firmware or hardware changes to support a new protocol or standard.

Building a side-channel radio that only relies on a processor is non-trivial and requires addressing multiple challenges. The fundamental challenge is that EM side-channels generated solely by the processor are often much weaker than the other types of side-channels [7]. To tackle this, we propose creating a modulation technique based on Chirp Spread Spectrum (CSS) technology to boost signal strength. In particular, by executing a specific sequence of instructions in software, we create CSS and show how it enables a LoRa-based communication protocol. Note that our solution is purely based on software and enables a microprocessor to directly send data to a receiver without using any radios or antenna. Moreover, our method is applicable to commercial-off-the-shelf (COTS) microcontrollers without any modification to them.

Figure 1 shows the overview of a conventional communication system in comparison to our design (called Side-Channel Communication or SideComm). As shown in Figure 1a, in a conventional communication system, a microprocessor is connected to an RF module (e.g., BLE, WiFi, LoRa) to send data packets[1]. Alternatively, as shown in Figure 1b, we propose leveraging the unintentional RF radiation created by the microprocessor as a means for sending RF packets. To follow a specific protocol (e.g., LoRa), a piece of specialized software needs to be executed on the microprocessor (§4). Our design eliminates the need for conventional radios and antennas. In particular, the wires and transistors on the processor and the board collectively act as an antenna and radio. Further, the RF signals created are the direct result of the computation and no additional RF front-end or baseband processor is needed. Thus, SideComm can provide a flexible and easily implementable solution for IoT communication.

We envision that our proposed communication method is used in scenarios where *low-rate and close-by (<10 m) data transmission is needed*, while the node is severely resource-constrained. These constraints could be in the form of power consumption, cost, form factor, and computational power. Our vision is that our system could be an excellent candidate for "smart" environments (e.g., smart homes, smart factories, hospitals, etc.) where a collection of various low-power IoT devices with different designs and applications are deployed in proximity (e.g., 10-15m) or within adjacent rooms or hallways. Using our method enables connectivity for these IoT devices and allows them to transmit their sensing data to a gateway or an access point.

We implement our method on two different microcontrollers: a TI MSP430 and Arduino Uno. For the receiving device, we implement a LoRa receiver that uses a low-cost antenna and a software-defined radio (SDR). We evaluate our system under different conditions including various distances, non-line-of-sight, and through-the-wall measurements (§5).

Contributions. While the idea of leveraging unintentional RF radiations, i.e., *analog side-channels*, has been extensively explored in the past as a security vulnerability to launch covert and side-channel attacks [17, 18, 41], ***our new contribution is to utilize side-channels purely from the processor itself as a means to enable low-power, low-cost and flexible communication***. Our solution enables a simple microprocessor to communicate directly with a receiver without using any radio (i.e., eliminating antennas, RF front-end, baseband processors, etc).

Specifically, the contributions in this work are:

- We propose a new method that leverages side-channel RF radiations *from the processor* to enable low-overhead communication for IoT devices without complex memory systems.
- We propose a new modulation technique purely controlled by software based on the *CSS* protocol for our side-channel communication. Our solution does not need any additional RF modules.
- We implemented our design in commercial-off-the-shelf (COTS) resource-constrained embedded/IoT devices such as the Arduino Uno and MSP430 series without any modification to them.

The remainder of this paper is organized as follows. In §2, we review important topics in side-channels and LoRa communication. In §3, we describe the need for our design as well as the fundamental relationship between software activity and the corresponding side-channel signals. We present our transmitter design and our microbenchmark for creating the desired side-channel activities in §4. The receiver design is also explained in this section. Our setup and results are presented in §5. Related work is reviewed in §6. The paper is concluded in §7.

[1]The processing and RF module could be integrated into the same circuit (i.e., an SoC) or they could be two separate boards

2 BACKGROUND

Side-Channels. Computing devices create digital and/or analog signals that are often unwanted artifacts of the computation. Known as side-channel signals, these signals convey information about the code and data being executed on the hardware.

Side-channels are largely considered vulnerabilities, as devices can leak sensitive information such as cryptographic keys via side-channels. However, more recently, there has been a growing interest in utilizing side-channels for useful purposes, such as debugging, malware detection, and Trojan detection [19, 21, 56].

There are various types of side-channels, including timing, microarchitectural, power, and electromagnetic (EM). Among those, EM is particularly interesting in the context of communication and this work, since EM side-channels can be received from a distance similar to other types of radio frequency (RF) signals. EM side-channels are created due to the switching activities of the transistors and other components in a circuit and can be measured in various ranges from a few centimeters to meters or even tens of meters away using custom-designed receivers [9, 37].

Studies have shown that EM side-channels exist in various frequency ranges, however, the strongest ones are typically created by three main sources: the power management unit (PMU), the processor's clock circuitry, and the memory clock [7]. These three sources are particularly powerful since ***(i)*** the majority of the device's power consumption is dissipated in these activities, and ***(ii)*** they are very deterministic and hence create strong *periodic* signals. This is specifically helpful when frequency analysis is used, where these electromagnetic side-channels appear as "spikes" at the frequency, $f_c = 1/t_{period}$, and multiple *harmonics* of this fundamental spike, i.e., $\pm 2f_c$, $\pm 3f_c$, etc. We will use this fundamental observation to create our communication protocol as described in §3.

LoRa Communication. Long-range (LoRa) communication is a wireless technology that utilizes a chirp spread spectrum (CSS) to enable applications that need extended range [14, 33]. Typical wireless technologies such as WiFi have poor receiving sensitivity at longer distances. LoRa enables long-range applications because the signal is spread over frequency and time, allowing for greater robustness on the receiver side and resistance to attenuation.

Since we will use the LoRa protocol in this work, there are a few fundamental LoRa terms that need to be defined. A symbol sent in LoRa is a chirp. A LoRa chirp linearly sweeps a predefined bandwidth (BW) and represents a sent value [27, 48]. The spreading Factor (SF) defines the possible values and thus the number of bits that can be sent per chirp. There are SF bits contained within a chirp, with 2^{SF} possible values [3, 14]. A chirp is made up of 2^{SF} chips [3], which are smaller subsections of the linearly swept frequency. The chip rate is equivalent to the bandwidth [42]. The base chirp mathematically is represented as $Chirp(t) = e^{j2\pi t(f_0 + \frac{BW}{2T}t)}$, where T represents the chirp duration and t is the current timestamp. f_0 is the base frequency [41].

To send a particular value, one applies a cyclic time shift to the base chirp that allows for the encoding of different bits [14, 27], essentially a shift in the initial frequency [48]. To cyclically time shift the chirp, we multiply the base chirp equation by an additional term based on the desired time shift ΔT.

$$Chirp(t) = e^{j2\pi t(f_0 + \frac{BW}{2T}t + (1 - \frac{\Delta T}{T})BW)}, \quad 0 \le t < \Delta T. \tag{1}$$

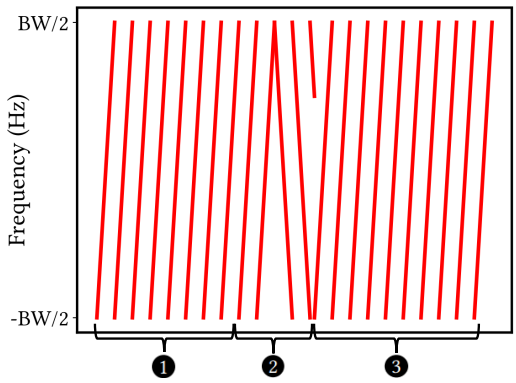


Figure 2: LoRa packet structure: 1) Preamble. 2) Sync Word and Synchronization Chirps. 3) Payload.

$$Chirp(t) = e^{j2\pi t(f_0 + \frac{BW}{2T}t - (\frac{\Delta T}{T})BW)}, \quad \Delta T \le t < T. \tag{2}$$

For a cyclic shift of $\frac{T}{4}$, the chirp starts at $\frac{BW}{4}$ and linearly increases to $BW/2$, from $0 \le t < \frac{T}{4}$. Then, at $t = \frac{T}{4}$, the chirp wraps around to f_0 ($-BW/2$) and linearly increases back to $\frac{BW}{4}$ from $\frac{T}{4} \le t < T$.

To demodulate a LoRa chirp, the receiver multiplies the received chirp by the conjugate of the base chirp (essentially the base downchirp) in the time domain [3, 27], as shown in Equation 3. Multiplication in the time domain is equivalent to convolution in the frequency domain. By taking the FFT of this product, we will obtain a spike correlating with the chirp value sent.

$$Value = Chirp(t) * \overline{BaseChirp(t)}. \tag{3}$$

LoRa follows a particular packet structure, as shown in Figure 2. First, the LoRa transmitter sends a preamble of known bits such that the receiver can synchronize and correct for carrier frequency offset (see ❶) [14]. Next, the LoRa transmitter sends two upchirps, two downchirps, and one-quarter downchirp (see ❷). The two upchirps are used to convey the network identifier called the sync word, whereas the two-and-a-quarter downchirps are used for frequency synchronization, also called the start frame delimiter [27]. Finally, the LoRa transmitter sends the payload (see ❸), which is the actual data being delivered.

In the next two sections, we first describe how and why side-channels can create a chirp modulation, and then in §4, we present the details of our LoRa protocol.

3 OVERVIEW

3.1 The Need For a New Design

To motivate the need for a new method of communication, we first highlight the main differences between our proposed method and state-of-the-art.

For communication in resource-constrained IoT devices, conventional transmitter modules (e.g., WiFi and BLE) are standard. These transmitter modules are typically an *additional* expense in terms of hardware, monetary value, flexibility, and energy. Our proposed method, on the other hand, eliminates the need for having an additional communication module and *reuses* the existing circuitry and computation for transmitting messages. This is beneficial in

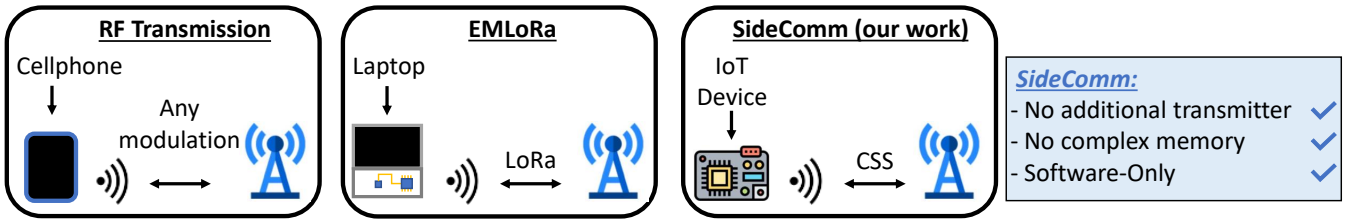


Figure 3: Comparison of communication via RF transmitter, memory accesses, and processor code. SideComm doesn't need an additional transmitter module or a complex memory system and creates CSS via simple code that AM modulates the clock.

Table 1: Comparision of SideComm compared to existing works. SideComm works even for devices that do not have DRAM.

Device	EMLoRa [41]	Noise-SDR [8]	SideComm
Arduino Uno	✗	✗	✓
TI MSP430	✗	✗	✓
Argon	✗	✗	✓
Feather M4	✗	✗	✓
nRF5340 DK	✗	✗	✓
Omega2Pro	✗	✗	✓
Particle Boron	✗	✗	✓
STM 32	✗	✗	✓
Beaglebone	✓	✓	✓
ESP32	✓	✓	✓
Raspberry Pi	✓	✓	✓

situations where a device cannot support a transmitter module, or we do not want to incur the added cost, especially for low-energy low data-rate scenarios.

Apart from conventional RF transmitters, very recently a new method based on leveraging EM emanations has been proposed [41]. This method, called EMLoRa, leverages RF radiations from the DRAM and the memory clock to infer information (including using it for covert communication). The major caveat of EMLoRa (and another similar work called Noise-SDR [8]), is that **these approaches are only applicable for devices with DRAM**, such as a laptop, *while not being applicable for low-end IoT devices.* The reason is the signal in EMLoRa and Noise-SDR is modulated via DRAM accesses. SideComm, however, *removes* the need for complex memory, potentially enabling microcontrollers and embedded systems that do not have complex memory systems, but rather simple memory systems such as flash and SRAM, to communicate. The challenge in SideComm, however, is to find a new technique that does not rely on complex memory accesses to create robust signals for IoT devices.

We summarize the key differences between our new method, SideComm, and state-of-the-art in Figure 3, and emphasize that **a new design that is suitable for low-end resource-constrained devices** is needed to enable low-overhead and flexible communication for IoT devices. In Table 1, we also provide several common IoT devices and describe whether or not EMLoRa [41], Noise-SDR [8], or SideComm will work on those systems.

SideComm enables IoT nodes to communicate with an access point without requiring any radios. In particular, IoT nodes use their processor's side-channel emissions (SideComm's transmitter) to send their data to the access point (SideComm's receiver). SideComm is based on two underlying phenomena. First, an *AM* modulation happens due to the fundamental relation between periodic software activities and the underlying hardware that creates side-channels. Second, a side-channel chirp signal can be created by an intentional software activity that continuously changes the duration of periodic activities. Using these two phenomena, a LoRa communication protocol can be implemented. In the following, we will describe these two fundamental events in more detail. In §4, we describe SideComm's transmitter and receiver designs.

Step 1) Side-Channel AM Modulation. In §2, we briefly described that analog side-channel signals, e.g., electromagnetic emanations (EM), are created due to changes in the current. More concretely, instructions within a program can consume various amounts of current, and hence a unique analog pattern is created when a piece of software is executed. Such a pattern can then be received as an RF signal through EM side-channel emanations. Since these patterns are correlated with the code and data being executed, repetitive execution of the same set of instructions would result in (very) similar RF signals. Moreover, the clock activity of the device itself also creates a very repetitive activity. Due to the periodic nature of these two groups of activities, one can analyze the created signals in the frequency domain, where each repetitive activity appears as a "spike." An example of these periodic activities can be seen in Figure 4a and b. In each figure, the time-domain signal is shown on the top. The corresponding frequency-domain signal is shown at the bottom (shown as a *spike*).

Interestingly, an unintentional AM modulation is created when repetitive software activities are combined with repetitive circuit activities (i.e., the clock circuitry) [39]. Again, recall that various instructions consume different amounts of current/power. Moreover, all instructions are synchronized with the clock. These combined induce an AM modulation where the clock is the oscillator (carrier signal) and is mixed with the periodic software activity (i.e., the message). This is shown in Figure 4c. Both sidebands can be seen in the figure, however, only one is highlighted for clarity.

Step 2) Creating Chirp Signals in Software. Building upon the observation in *Step 1*, our key insight in this work is to leverage these "spikes" created by repetitive software to make a controlled *chirp*. The key idea is to use a *nested* loop, where the frequency of the outer loop is essentially controlled by the number of iterations of the inner loop. Connecting this to the AM modulation described earlier, the outer loop is similar to what is shown in Figure 4b. The

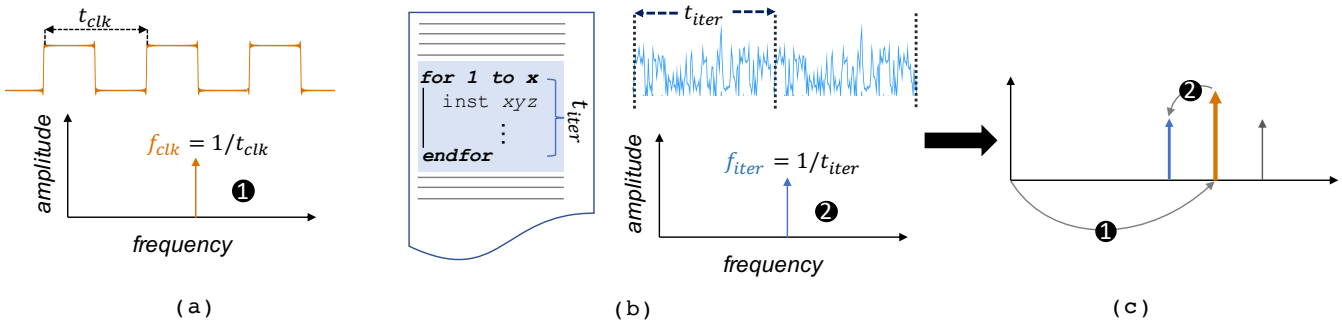


Figure 4: Periodic activities, such as FOR loops (b), that are synchronized with the clock (a), can create an AM modulated analog side-channel signal (c).

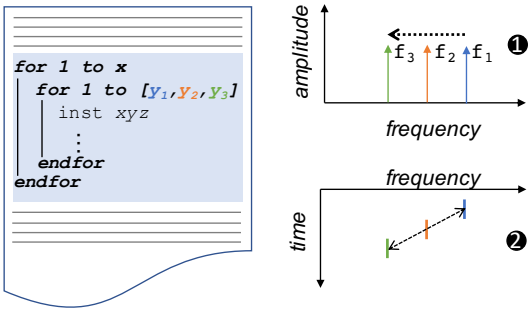


Figure 5: By changing the duration of a periodic activity (e.g., a FOR loop) a *chirp* can be created.

difference is that its duration is no longer fixed and is modulated by the inner loop. Thus, changes in the inner loop count change the *frequency* of the AM-modulated sideband signals.

This observation is shown in Figure 5. As can be seen, instead of having one loop, a nested loop is created where the iterations in the inner loop change by controlling a variable, y. For example, in Figure 5 (see ❶) we show how by changing y to three different numbers, three different signals at f_3, f_2, and f_1 can be created (we assumed that $y_1 < y_2 < y_3$). This roughly creates a chirp when plotting the spectrogram (frequency over time, as shown in ❷).

3.2 Fundamental Techniques for SideComm

The important takeaway is that such a chirp is completely controlled by software activity, and the resulting EM side-channel can be received from some distance (details in §5). *No RF circuitry (analog or digital) is needed* to create the desired modulation, which could potentially save a lot of power. While encouraging, there are several challenges to creating a reliable and low-power communication protocol. In the next section, we describe how we carefully designed a transmitter and a receiver that utilizes these phenomenons.

4 SYSTEM DESIGN

To create a side-channel-based communication system, a functional transmitter and receiver need to be designed. In this section, we will first discuss the transmitter, which consists of a device emanating LoRa-style chirps by adjusting the frequency of a repeated segment of code. Then, we will discuss the receiver. Lastly, we will discuss how the transmitter and receiver communicate with each other.

4.1 SideComm's Transmitter

As mentioned earlier, SideComm enables IoT nodes to reuse their microprocessor as a transmitter without requiring any antenna, RF front-end, or digital baseband. SideComm's transmitter has several components which are described in the following.

1) "FOR" Loop Code. As discussed in §3, repeated phases of the code amplitude modulate the clock, resulting in activity in the sideband of the clock. For instance, for a clock at frequency f_c and code repeating at a frequency of f_l, spikes are seen in the frequency spectrum at $f_c + f_l \cdot n$ and $f_c - f_l \cdot n$, where n refers to the specific harmonic. This property can be utilized in chirp emulation. Instead of maintaining a static loop frequency, the loop frequency can be adjusted to create a chirp.

Algorithm 1 displays the pseudocode for the side-channel LoRa transmitter. The innermost FOR loop (**Step 4**, **lines 4-6**) is the code that can be adjusted to emulate a chirp. We devise a set of frequency iterations, *FreqIterations*, that can be swept in time to create a chirp. For instance, increasing *FreqIterations* will increase the time it takes for the FOR loop to complete, decreasing the frequency, whereas decreasing *FreqIterations* will decrease the completion time, increasing the frequency. For a given spreading factor, a range of *FreqIterations* of an amount equal to 2^{SF} is defined, which controls our chirp bandwidth.

To understand the observed chirp frequency from a given *FreqIterations*, we need to analyze the execution of compiled assembly instructions and understand the number of instructions in a single iteration of our FOR loop and the average cycle time elapsed during these instructions. For example, in one setup, we use an *Arduino Uno* which has 29 instructions in a single iteration of our innermost FOR loop and an elapsed time of 52 cycles. Given this information, we can obtain the frequency for a given *FreqIteration* as $\frac{ClockFreq}{LoopCycles \cdot FreqIterations}$. This resulting frequency is a function of the clock frequency *ClockFreq*, the number of clock cycles observed *LoopCycles*, and *FreqIterations*.

2) Chirp Emulation. Utilizing the frequency-shifting capability, we can emulate LoRa chirps. In the classical LoRa standard, chirps

consist of a linear change of frequency over time. Each chirp consists of many chips. In our LoRa emulation, however, we are limited by the frequency steps that can be created by our innermost FOR loop. Thus we emulate a LoRa chirp by cycling through steps in frequency that, when combined, emulate a chirp. These individual steps are the chips of the chirp.

The number of steps is defined by the spreading factor chosen. For instance, a spreading factor of 4 would result in 16 steps (2^{SF}). The resulting chirp bandwidth is defined by the frequency differences instilled by changes in *FreqIterations*. Given Equation 4, we can derive the bandwidth given a min/max *FreqIterations*, where a smaller *FreqIterations* gives a higher frequency.

$$BW = \frac{ClockFreq}{LoopCycles} \cdot \left(\frac{1}{FreqIterMin} - \frac{1}{FreqIterMax}\right). \quad (4)$$

Algorithm 1: Side-Channel Transmitter Code

```
Data: value to send: Val
Result: chirp of desired value transmitted
// Step 1: set starting frequency to begin sweep
1  FreqIterations = FreqIterMax − 1·Val
   // Step 2: Send each chip
2  for i=0 to NumChips do
       // Step 3: Chip dwell time
3      while j < ChipDurationCount do
           // Step 4: Chip frequency control
4          for k=0 to FreqIterations do
5              sum = sum + 1;
6          end
7          j = j + 1;
8      end
       // Step 5: Update to next FreqIteration
9      FreqIterations = NextFreqIteration;
10 end
```

To send a chirp, we first need to be able to emulate a *chip*. To create a chip, we maintain a given *FreqIterations* for a period of time, the dwell time, equal to $\frac{1}{BW}$ (encoded into the value of *ChipDurationCount*). We repeat the innermost FOR loop until this time has elapsed. This is the WHILE loop in Algorithm 1 (**Step 3**, starting at **line 3**). Finally, the outermost FOR loop in Algorithm 1 (**Step 2**, starting at **line 2**) iterates over the number of chips in our chirp, *NumChips*. For instance, the outermost FOR loop will cycle through 16 iterations, for 16 different *FreqIterations*, given a spreading factor of 4. The starting value of *FreqIterations* is set in **Step 1**, **line 1**, where we initialize *FreqIterations* to its initial value. *FreqIterations* is updated to the next value needed for the chirp in **Step 5**, **line 9**. For a chosen set of parameters (e.g., *FreqIterations* from 135-150), and Equation 4, the guard band and chirp bandwidth can be calculated.

3) Sending Packets. To send a packet, we need to send a stream of chirps that follow the LoRa standard. In a LoRa packet, as shown in §2 and Figure 2, we have a combination of various up chirps and down chirps. The preamble is sent by sending a predetermined amount of consecutive base chirps (in our case, 8 upchirps). The sync word is sent by sending the correct values that represent the given sync word. For instance, the sync word of 0x11 results in two upchirps of value 8. Then, the start frame delimiter is sent, which is two consecutive downchirps and a quarter downchirp. Finally, the payload is sent, where upchirps of various values can be sent to transmit the payload (data).

When not transmitting, our code executes an idle code. For a real-world scenario, we can assume the code will transfer the control back to the firmware and/or [main] function when the transmission has been completed.

4) Challenges. An interesting challenge is the frequency jumps observed in our LoRa emulation when changing *FreqIterations*. As explained earlier, the smallest change in frequency is determined by a single change in *FreqIterations*, as *ClockFreq* and *LoopCycles* are assumed constant during calculation.

We are also emulating a chip with a constant frequency step. This means that frequency jumps exist per change in *FreqIterations*. Large frequency jumps, or taller steps, are not desired as this will increase the difficulty of demodulating the chirps. On the other hand, a set of *FreqIterations* with a small frequency jump at each step is desired. This increases the smoothness of the chirp as well as the SNR. In fact, the executed code has slight variations in frequency and time, resulting in an averaging that creates greater smoothing within a chirp.

Another interesting consideration is the dwell time. Generally speaking, maintaining a constant frequency for longer will increase the signal power. LoRa defines step time as $\frac{1}{BW}$, where BW is defined by *FreqIterations*. We picked a set of *FreqIterations* where the step time was long enough to see the relevant signal strength.

An additional challenge faced was the fact that chirps in near proximity to the clock are less visible due to interference. This can be fixed by increasing the guardband of the chirps by setting the base frequency of the bandwidth to be higher. If this can be done without sacrificing the smoothness and strength of the chirp, one should do so. Additionally, one can measure the signals at higher harmonics where the interference is minimal. We will discuss this further in §4.2.

4.2 SideComm's Receiver

SideComm enables devices' microprocessors to directly send their data to the air. For an access point to receive this data, we need to design a receiver that can demodulate side-channel LoRa chirps. We design and implement our receiver using a software-defined radio (SDR). We utilize the GNU Radio package gr-lora-sdr [47]. We modify the code where needed. The rest of this section discusses the various components of the receiver and the modifications needed to successfully demodulate. The block diagram of our receiver is shown in Figure 6.

1) Finding the Frequency Band. The first step in the receiving process is finding the correct frequency band. Since the transmitter relies on the clock as the carrier, the transmission will be centered around the clock at frequencies $f_{clk} \pm f_{chirp}$. Interestingly, we also observed that analog EM emanations create strong harmonics, mainly due to the square shape of the carrier (clock) signal, as also validated by prior work [7]. This means that the transmitted signal

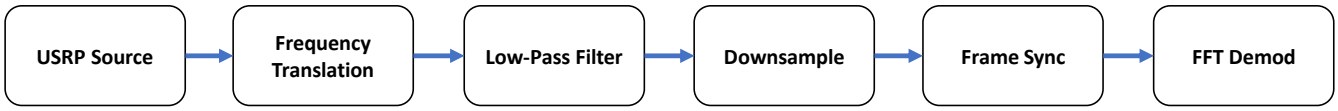


Figure 6: Block diagram of our side-channel receiver. i) USRP Source: obtain data from SDR. ii) Frequency Translation: center the frequency on our chirps. iii) Low-Pass Filter: remove unwanted upper frequencies before downsampling. iv) Downsample: reduce the sampling rate of the data. v) Frame Sync: controls preamble detection, sync word, and frequency correction. vi) FFT Demod: demodulate the symbol.

also exists in multiple higher harmonics of the original signal, i.e., at $m \times f_{clk} \pm f_{chirp}$, where m is the mth harmonic of the base signal.

Based on the availability of strong higher harmonics, the receiver has a degree of freedom to pick a channel (i.e., *mth* harmonic) to maximize the performance. We observed that the receiving signal strength is a function of *two* parameters. ***(i)*** *Channel occupancy/interference* and ***(ii)*** *circuit frequency response*. The first is depending on the spatial and temporal features of the environment, similar to any other conventional communication system. As a result, to maximize the performance considering only the first parameter, it is best to pick a harmonic/channel with the lowest interference. The receiver can dynamically sense the channel and pick the best harmonic/channel. It is important to mention that the transmitter does not need to change anything, nor does it need to be aware of which channel is being used (the harmonics are automatically and unintentionally generated).

The second parameter, however, is *device-specific*. The main reason is that, unlike conventional RF systems with a well-designed and optimized antenna, the antenna used in our setup is the circuit itself - i.e., our design is antenna-less in the sense that the electronic components of the device collectively create the EM analog side-channel signals. We further investigated this and found that similar to any conventional antenna, depending on the material, layout, and design of the circuit board, each board has a unique frequency response. Figure 7 shows the frequency response of the Arduino Uno board, a popular microcontroller device that is used in our setup. The figure shows the signal-to-noise ratio (SNR) of observed chirps, measured in dB using a magnetic probe placed next to the device. We performed the measurement for different channels swept from 112MHz to 656MHz[2], with a higher density of channels selected around 448MHz. We see that the SNR varies depending on the given channel. For instance, the SNR at 304MHz and 592MHz is quite low (7dB), whereas some channels, such as 400MHz and 448MHz, have a comparatively higher SNR of 27dB and 25dB, respectively. The majority of the channels have an SNR of around 15-20 dB.

Given these two parameters, the receiver picks a center frequency and adjusts the filtering and other preprocessing functions, which are described in the following. Again, no changes are needed on the transmitter side.

2) Filtering and Downsampling. With the correct center frequency, we now need to filter and downsample the signal for demodulation purposes. We first apply a low-pass filter in order to reduce aliasing before we downsample. Then, we utilize rational

[2]We did not investigate ranges below 100 MHz to make the receiver's antenna size compact and manageable. Details of our setup are given in §5.1

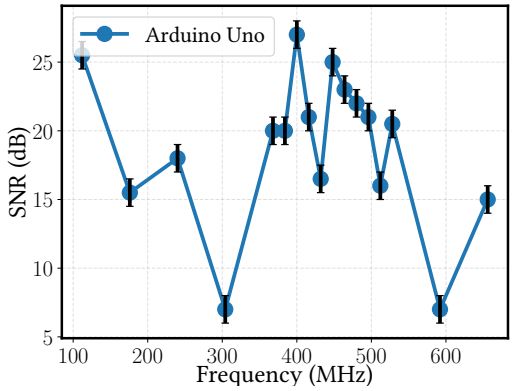


Figure 7: Chirp SNR created by the microprocessor side-channel signal for various frequencies, swept from 112MHz to 656MHz, using a magnetic probe. The SNR varies depending on the frequency.

resamplers to downsample our data from the base sampling rate (e.g., 500 kHz) to four times our chirp bandwidth. The factor of four is used to be able to properly demodulate each chirp and is a design choice made by gr-lora-sdr [47].

3) Frame Sync and Demodulation. As in any LoRa receiver, we detect packets via preambles and utilize the preamble to estimate and correct for carrier frequency offset. We then demodulate the sync word as the network identifier. If the network identifier matches, we then process the two-and-a-half-quarter downchirps to synchronize and discern the relationship between the sampling time offset and the carrier frequency offset. Finally, we demodulate the payload.

Similar to prior work [30], we can utilize a reduced rate mode to increase robustness. For a spreading factor of 4 (16 values), we can divide our chirps into four separate bins (15-2, 3-6, 7-10, 11-14) in order to reduce the demodulation errors.

4) Challenges. The major challenge in our demodulation is the *frequency shifting* phenomenon. As the transmitter device heats up, the frequency of the clock will (slightly) shift. This small clock shift will affect the AM modulation of the clock as well. Thus, we need to adjust the center frequency dynamically on the receiver side to compensate for this shift and be able to demodulate in the correct region in the frequency spectrum.

To compute the shift on the receiver side, we take an FFT over our signal (using the expected center frequency and range) and find the index with the maximum value. This index should show us

the new position of the clock. After converting this index into the frequency, we can then use it, plus the given offset of our LoRa chirp and bandwidth, to compute the new center frequency of our chirps. For example, in our setup, we found that adding a maximum of 2.117 kHz to the real frequency of the clock is sufficient to find the center of our chirps. This frequency correction happens periodically over time (e.g., once every 10 seconds), hence the synchronization only needs to happen rarely without interfering with too many packets.

5 EVALUATIONS

5.1 Experimental Setup

To evaluate the efficacy of SideComm, we design a range of experiments using multiple different boards. Specifically, we use two widely popular microcontroller boards, *Arduino Uno* [50] and *TI MSP430G2ET* [34]. ***We make no changes to the hardware configuration.*** The hardware is used as-is out-of-the-box, and the signal strength was not boosted or modified in any way. For Arduino, we use a 16MHz clock and for MSP430, a 12.5MHz clock is used. As described in §4, we measure both devices' signals at higher harmonics. For Arduino, we measure the signal at the 28th harmonic, i.e., 448MHz. For MSP430, we measure at 400MHz, the 32nd harmonic.

For the receiver, we use a COTS VHF/UHF indoor TV antenna [4] connected to a *USRP B205mini-i* software-defined-radio [5]. We explicitly chose a cheap antenna to highlight that we can achieve decent results even when using such a setup. For all measurements, we set the internal amplifier of the SDR to 15 dB. For the receiver, we modify the gr-lora-sdr package [47] on GNU Radio. Unless indicated otherwise, we use a 500kHz sampling rate with 500kHz bandwidth to collect the RF side-channel signals in our measurements.

An example of our measurement setup, including the transmitter device and the receiver setup, can be seen in Figure 9 (line-of-sight measurement for distance equals 5m). If not specified, all measurements are conducted indoors to emulate the potential target scenario (i.e., smart homes, factories, hospitals, etc.). The measurements are all conducted in a research lab and/or its surrounding hallway, located in a metropolitan area with lots of interference. We do not attempt to reduce or shield any outside signals.

Furthermore, to minimize interference, we observe that each device has a unique clock frequency due to imperfections in the manufacturing process. Thus, if multiple devices are in the same room and are either 1) communicating via SideComm or 2) conducting normal operations, there should be enough separation in the spectrum. However, if there is contention in the band, we change the parameters of our modulation loop (such as $FreqIterations$) to separate our chirps in the band.

The TV antenna is attached to the side of a movable cart. The test board is attached to a tripod that sits on a second movable cart. This way we can rearrange the antenna and board for different test scenarios. Additionally, the height of the tripod and the angle of the board were adjusted to be reasonably aligned to the antenna (note that this setup can be further fine-tuned to increase SNR. For our tests, we provided a rough alignment).

To transmit bits, we use the code shown in Algorithm 1. We implement the code for both boards using Arduino and Code Composer Studio IDEs. For both boards, we hand-optimize the code by implementing it on assembly. We further disable compiler optimization to fully control the timing of chirps (we observed that the compiler might optimize the code through loop unrolling and/or other methods, which would result in an unwanted change to the loop's iteration time). After all optimizations, the implementation of Algorithm 1 on Arduino results in 283 static instructions. For MSP430, there are 91 static instructions.

5.2 Results

First, we will discuss various SNR measurements of SideComm in different scenarios. Then, we will present the bit error rate and bit rate results. Finally, the power and energy consumption of SideComm and a comparison with other methods are reported.

An example of the received signals using our setup is shown in Figure 8. We see the spectrogram with real chirps created by our microcontroller (Arduino) measured by our software-defined radio (in 2m distance). The value "8" is sent, then the value "0", in the spectrogram. We see the clock at the center of the spectrogram, at 0kHz offset. We see the central chirps at a frequency relative to the center of -2kHz and 2kHz, a set of mirror images (for this example we chose the offset to be 2kHz, however, this could be controlled by the user). Then, at -4kHz and 4kHz, we see the second harmonic (higher harmonics are not shown).

SNR is calculated by measuring the strength of the chirp signal and the level of the noise floor over an extended period of time. In this way, we can account for temporal variations seen in both the emulated signal and the background noise by averaging.

1) Indoor, Line-of-Sight SNR. We see the results of SNR vs. Distance for an indoor line-of-sight scenario for Arduino and MSP430 in Figure 10. At a close range of 2m between our SideComm transmitter and receiver, Arduino has an average SNR of 10dB while MSP430 has an average SNR of 15 dB. At 5m, Arduino has an SNR of 8dB while MSP430 has an SNR of 14 dB. Finally, at 15m, Arduino has an SNR of 3dB, while MSP430 has an SNR of 8 dB. We repeat these measurements 5 times and report the error in the figure. Overall, we observe that the EM emanations of MSP430 are consistently stronger than those of Arduino. This demonstrates SideComm's ability to propagate over long distances, as we still measure adequate SNR even at longer ranges. Additionally, LoRa is shown to work when the signal is below the noise floor, further suggesting the validity of the signal received.

We did not extensively explore the signal reception beyond 15m, however, our initial investigation and the robustness of LoRa-based receivers to noise suggest that the range could be extended even further. However, we note that this assumption is anecdotal and further experiments are needed to confirm it. One can also further optimize the setup and directionality of the transmitter and receiver to further boost the SNR. We did not explore this further, since our primary goal was to show the feasibility of this approach in reasonable distances without requiring fully optimizing the transmitter and/or receiver and with minimal engineering work.

We see the SNR measured for the Arduino and MSP430 for these two different NLOS scenarios in Figure 13. The point-to-point distance between the microprocessor and the antenna (receiver) is roughly the same for both setups.

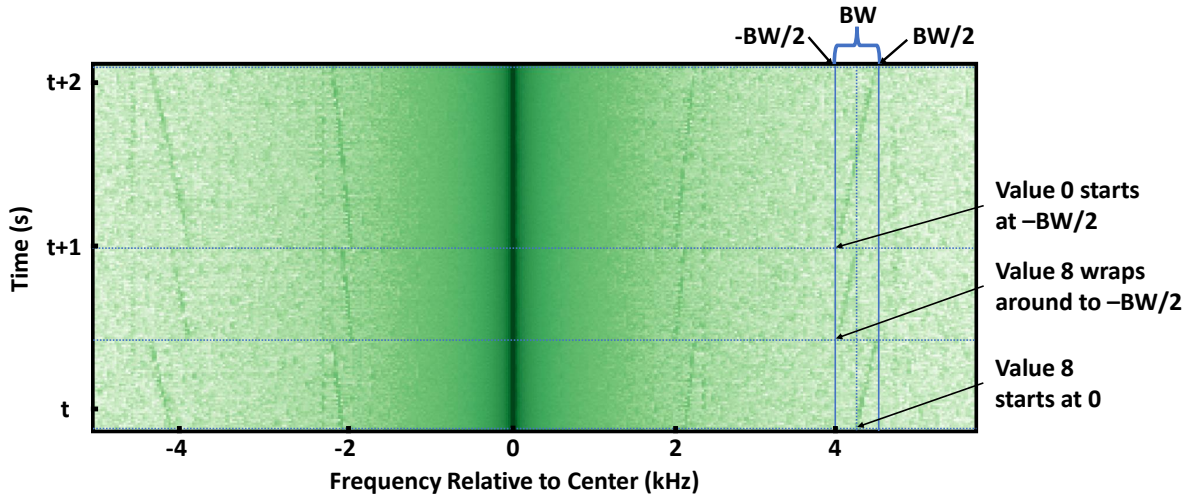


Figure 8: Chirps viewed via a spectrogram. Values sent: 8, then 0. The clock is at a frequency offset of 0kHz. The central chirps are at a frequency offset of -2kHz and 2kHz. The 2nd harmonic chirps are at a frequency offset of -4kHz and 4kHz. Higher harmonics are not shown.

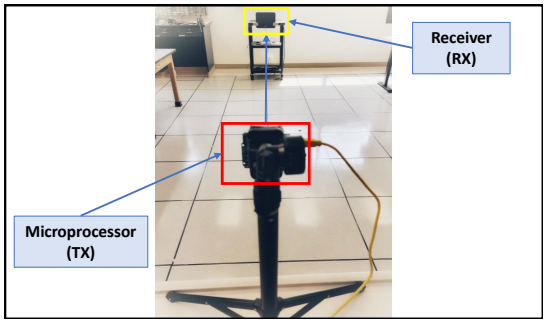


Figure 9: Indoor, line-of-sight setup. The setup in this picture has the microprocessor (Arduino Uno) and the receiver (small TV antenna and SDR) 5m apart.

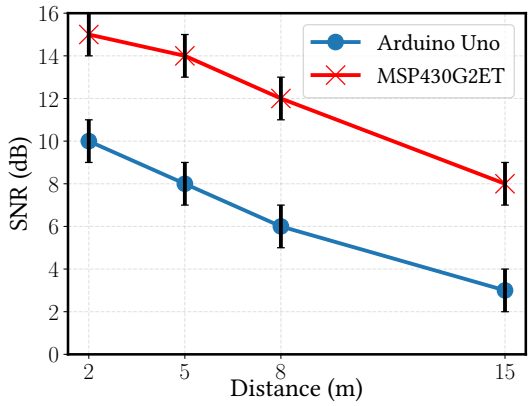


Figure 10: SNR vs distance, indoors, line-of-sight. The SNR was measured at 2m, 5m, 8m, and 15m. The MSP430 consistently has a higher SNR than the Arduino.

For the "around the corner" scenario, the Arduino has an average SNR of 3.5dB, whereas the MSP430 has an average SNR of 6.5 dB. For the "through the wall" scenario, the Arduino has an SNR of 6dB, whereas the MSP430 has an SNR of 7 dB.

These results demonstrate SideComm's robustness towards environmental factors and resilience to sources of attenuation. SideComm can work in scenarios where the board is not lined up with the antenna or even separated physically. In the first scenario, we are able to receive the signal even when the board direction is perpendicular to the direction of the antenna and physically around the hallway corner. In the second scenario, the signal is also still seen despite the presence of an office wall between the board and the receiver. Robustness to environmental factors also enables many applications and boosts the adaptability of SideComm.

2) Indoor, Non-Line-of-Sight SNR. Beyond line-of-sight scenarios, it is important to test non-line-of-sight (NLOS) scenarios to test SideComm's ability to resist attenuation and environmental factors. We conduct two NLOS measurements, as shown in Figure 11 (around a corner) and Figure 12 (through a wall).

3) Outdoor, Line-of-Sight SNR. Beyond indoor measurements, we also conducted an outdoor line-of-sight experiment for Arduino and MSP430 to demonstrate robustness in outdoor settings. The setup for our measurements is shown in Figure 14. The results are shown in Figure 15. The results show that we are able to obtain a similar SNR to the comparable line-of-sight indoor scenario. Like our indoor measurements, the MSP430 still achieves a better SNR than Arduino. This outdoor experiment demonstrates the potential of using SideComm in various outdoor settings.

4) Bit Error Rate and Bit Rate. Beyond measuring the SNR of our side-channel chirps in various scenarios, we also tested the bit error

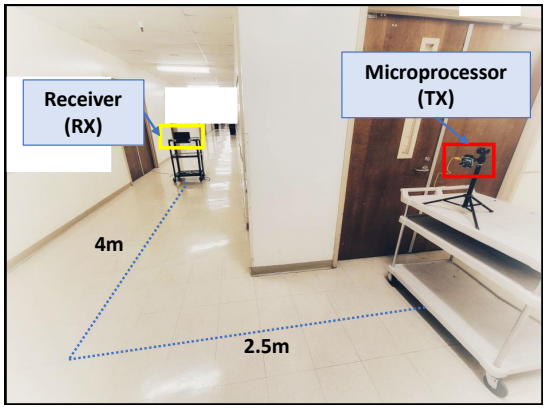


Figure 11: Indoor, non-line-of-sight setup: Around the corner. 2.5m from the microprocessor to the corner center, and 4m from the corner center to the receiver.

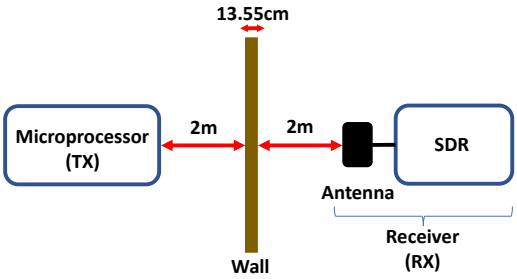


Figure 12: Indoor, non-line-of-sight setup: Through a wall. The wall is 13.55cm thick. 2m separates the microprocessor and the wall. 2m separates the wall and the receiver (antenna/SDR).

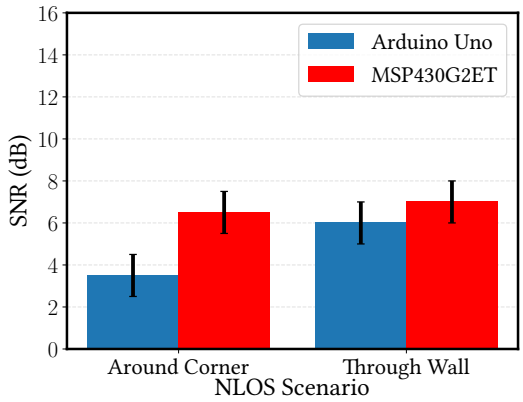


Figure 13: SNR for various non-line-of-sight scenarios. The SNR was measured when the processor and receiver were around a corner from each other, as well as when they were separated by a wall. Each setup has roughly the same point-to-point distance between the processor and receiver.

rate (BER) and the achievable bit rate we could obtain from our system using the receiver described in §4.2 and shown in Figure 6.

Similar to prior work [25], to test BER, we sent 30 packets, each with a 40-bit payload. This results in a total of 1200 payload bits. We then repeat this test in multiple trials and report the average using the setup described in §5.1.

We investigated various configurations and found that our setup can achieve up to 1kbps (to be exact, 997 bps for Arduino and 907 for MSP430) by reducing *FreqIter* in line 4 of Algorithm 1. This bit rate is sufficient for many IoT applications such as smart home sensors. For the maximum bit rate, the BER varies between 10^{-2} and 10^{-1} depending on the scenario and the distance between the node and the receiver. It is worth mentioning that this BER can be improved by reducing the bit rate. In particular, we observed that the bandwidth and length of the chips have an impact on the SNR and accordingly BER. The main reason for this is that more iterations in the inner FOR loop (see lines 4-6 in Algorithm 1), make the emanations stronger due to the averaging effect of taking an FFT. However, this comes with spending more time sending a bit, which consequently reduces the bit rate. Finally, one may use an error correction scheme to correct the errors and improve the BER.

5) Power Consumption. We measure the power consumption of SideComm's transmitter and compare it with several different systems and setups. Specifically, we compare our method with standard BLE, Zigbee, WiFi, and LoRa methods. When needed, we consider two possible scenarios: ***(S1)*** A network module (e.g., BLE) is connected to a microcontroller (e.g., Arduino); ***(S2)*** An SoC with a built-in microcontroller and network modules (e.g., TI CC2650 BLE chip [10]). Furthermore, we compare our method with state-of-the-art backscattering techniques.

To measure the power in our setup, an Adafruit INA219 [23] current sensor module is utilized. Each target board (e.g., MSP430) is then connected in series with the sensor. By measuring the voltage drop across a low resistance shunt, the INA219 module measures the current of each of the corresponding setups. To transfer the measurements to a PC, we use an Arduino connected to the INA219 module over I2C, and forward the current values to the computer over UART, sampling the current at about 7250 Hz.

To limit the number of tests, for other setups, we use their respective datasheet and/or publications to report the results. For each method, to provide a fair comparison, in all cases ***we report their minimum power consumption***. This is achieved by setting the lowest possible Tx power. We, however, acknowledge that the comparison for distance is not a fair one as SideComm could achieve around 10 meters while others could potentially achieve much higher distances. Nonetheless, we emphasize that this is the *minimum* power consumption for each method.

Moreover, for the sake of making it more favorable for other works, in each work, we only take into account the power consumption of the radio, not including the power consumption for the core, power management unit, and peripherals. If these had been taken into consideration, the actual power consumption would be higher. It is worth noting that, for our setup, we consider the total power consumption including core and non-core activities. The power consumption in SideComm, however, is the entire power (i.e., core and peripherals included). This means that we overestimate our

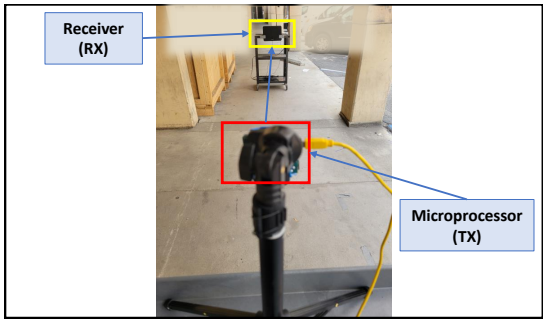


Figure 14: Outdoor, line-of-sight setup. The setup in this picture has the microprocessor (Arduino Uno) and the receiver (small TV antenna and SDR) 5m apart.

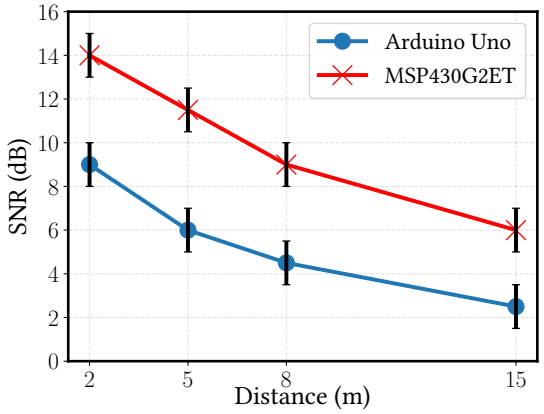


Figure 15: SNR vs distance, outdoors, line-of-sight. The SNR was measured at 2m, 5m, 8m, and 15m. A comparable SNR compared to indoors is seen.

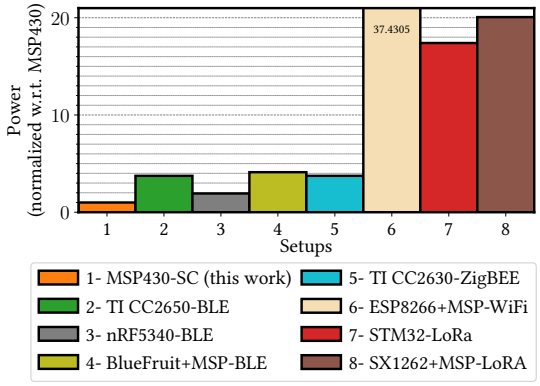


Figure 16: Power consumption for sending a packet in various setups. Values are normalized w.r.t. MSP430.

own method. This was specifically significant for Arduino as the overall was about 45x higher than MSP430.

Results for power consumption are shown in Figure 16. To better compare our method with others, the y-axis values are normalized with respect to MSP430 power consumption.

Comparing to LoRa devices (setup 7: STM32WL55JC [44] and setup 8: SemtechSX1262 [45]), MSP430 achieves about 20x lower power consumption. It is important to mention that these LoRa modules can achieve a much higher range than our method, however, for low/mid-range settings, our method offers a better power-distance tradeoff.

Comparing SideComm with BLE and Zigbee devices (setups 2-5) reveals that MSP430 has about 4x lower power consumption on average. The difference is higher when compared with a WiFi module (setup 6);

Putting it all together, SideComm achieves consistently lower power consumption compared to other active candidate technologies while eliminating the need for an additional RF module or an active transmitter module. It is important to emphasize that **SideComm is beneficial specifically for low data-rate scenarios** and is not effective for times where a high data rate is needed due to its low bit-rate.

6) Comparison with State-of-the-Art. Lastly, we compare SideComm with state-of-the-art in terms of applicability, distance, and overall energy. We compare three configurations: 1) An IoT+BLE where an MSP430 microcontroller is connected to a CC2650 BLE module [10] – a representative of popular configuration for low power communication in IoT market, 2) A Raspberry Pi with EMLoRa capability [41] as the latest side-channel based technology for communication, and 3) SideComm implemented on MSP430. The results are shown in Table 2.

We compare the three methods in five main categories. First, the **key advantage** of SideComm over EMLoRa is that it is applicable to low-end IoT devices while EMLoRa is only useful for systems with DRAM capabilities. In terms of distance, however, both EMLoRa and BLE outperform SideComm, indicating that our method is useful for indoor use cases. Compared to BLE, the key advantage of SideComm is that it does not need a dedicated radio, hence it offers flexibility, simplicity, and low overhead.

We further compare the overall energy consumption of SideComm with the state-of-the-art. For that, we assume a scenario where each day at least 1KB of data is needed to be sent while the device is in deep sleep mode (with some small leakage) when not transmitting. We then normalize the values and report them in Table 2. As can be seen, BLE has more than 218% larger energy consumption while **EMLoRa has several orders of magnitude higher energy**. SideComm has superior energy-per-bit consumption in low-bit rate scenarios mainly because it completely eliminates the need for additional RF circuitry and hence reduces the leakage. Furthermore, the simple package structure of SideComm greatly reduces the communication overhead in low-bit rate scenarios.

Finally, we compare the bit rate for the three configurations. As explained before, SideComm is the best option for scenarios where a low data rate is sufficient. Compared to EMLoRa, SideComm achieves three orders of magnitude higher bit rate while having

Table 2: Comparison between RF transmission (BLE), EMLoRa [41], and SideComm. SideComm enables IoT communication without the presence of a transmitter module and where complex memory systems are not present. The Energy per bit and Bit rate entries are normalized with respect to SideComm.

Approach	IoT	Radio	Dist (m)	Energy	Bitrate
RF (BLE) [10]	Y	Y	50-400	2.18x	≈1000x
EMLoRa[41]	N	N	70-250	≈1000x	.0012x
SideComm	Y	N	5-20	1	1

more than five orders of magnitude lower energy, making it by far the **best option** for communication in a low-overhead setting.

6 RELATED WORK

Side-Channels for Adversarial Purposes. Side-channels have been mainly considered a vulnerability and have been extensively analyzed to improve the security of systems. Side-channels can be created either digitally or physically. We mainly focus on physical side-channels since the digital side-channels (e.g., cache side-channels [54]) can only be measured within the device (in the digital domain) and hence are not relevant to the analog/RF domain.

Physical side-channels can be categorized into two main groups: side-channel attacks (SCA) and covert channel communication. The goal of an *SCA* is typically finding a secret value (typically a cryptographic key) by analyzing the physical side-channels [13]. Such a secret is assumed otherwise undetectable. For a physical *covert channel*, the goal is to extract a secret value from an air-gapped system—i.e., using a physical signal that can be measured externally, secret information can be extracted and communicated to the outside world. Such a signal could be temperature, variation in power, electromagnetic (EM) emanations, etc. In this work, we mainly focused on EM and/or RF-related side-channels.

Much work has been done on using different modalities to create long-range EM/RF covert channels. Zhan *et al.* [55] showed that by creating specific memory access patterns, an EM covert channel can be created that can penetrate a concrete wall. Covert channels in air-gapped networks can also be realized via cellular frequencies [17] and Wi-Fi signals [16].

Recently, Screaming Channels [9] have been developed to enable the recovering of encryption keys at long distances by showing that EM emanations can be accidentally *leaked* into the RF module (e.g., WiFi). To enable this, however, an RF module (WiFi or BLE) is required (i.e., the RF already exists, so there is no need for additional side-channel radio anyway).

Also relevant to this work, NoiseSDR [8] was recently proposed. Using a method similar to that of EMLoRa, NoiseSDR showed that any arbitrary modulation (including LoRa) can be created using EM side-channels. Similar to EMLoRa, the method only works on high-power computing systems. Further, for creating chirps, the method needs to rely on spread-spectrum clocking, which is typically unavailable in tiny IoT systems.

To summarize, compared to existing physical covert channel techniques, *our method offers two main contributions*. First, a covert channel only focuses on creating a stealthy channel with low BER and/or high bit rate. In our ecosystem, however, power consumption and flexibility have also been considered. Second, none of the existing methods showed success in achieving long-range on tiny IoT systems, as opposed to ours.

Side-Channels for Beneficial Purposes. Recently, it has been shown that side-channels could be used for good purposes too. For instance, EM side-channels have been used to detect rowhammer attacks, a popular attack on DRAM [56]. EM side-channels can also be used to passively detect malware on existing devices [20, 38]. EM side-channels can also be used to profile program execution [39]. Compared to these works, this is the first work to propose using EM side-channels for communication. Specifically, we showed how to create intentional signatures to boost the range and increase the SNR, which is essential for long-range and reliable communication.

Low-Power Communication. Another area of relevance to our system is methods for enabling low-power communication. For instance, BLE has been shown to have comparatively greater energy efficiency compared to Zigbee [43] and WiFi. In BLE2LoRa, the authors developed a methodology to craft LoRa-style chirps in a BLE packet to enable cross-technology communication [30]. In Wile, a Wi-Fi compatible system was developed that has similar power consumption to BLE [1]. WiChronos was a new communication framework that improves energy efficiency [36].

Much work is being done to enable applications using LoRa. For instance, Xie *et al.* [53] developed a system using LoRa to enable sensing of human respiration up to 50m. Additionally, LoRa is being used in localization techniques in both indoor and outdoor scenarios [31]. Deep learning has been shown to increase the SNR of LoRa decoding [28]. Work has been done to increase efficiency and robustness in channels using LoRa such as reducing LoRa packet collisions [49], increasing link robustness [48], and developing carrier-sense multiple access [12]. Finally, a system that dynamically adjusts LoRa parameters to increase energy efficiency per environment has been developed [29].

Another emerging solution for reducing the communication power is using backscattering technology. In Wi-Fi Backscatter, the authors utilized the existing Wi-Fi infrastructure to support backscatter [25]. In Backfi, the authors improved the prior work by providing even greater range and throughput while maintaining energy efficiency [6]. In WiTag, the authors improved the practicality of backscatter by using properties of the MAC layer [2]. In NetScatter, the authors improved the scalability of backscatter by utilizing a distributed coding mechanism [22]. A symbiotic radio was also developed to create passive IoTs [32]. Backscatter can be used in low-power applications such as micro-implants in the human body [52]. Many papers detail the intersection of LoRa and backscatter to create a low-power form of communication. LoRa Backscatter was the first instance of wide-area backscatter that uses LoRa [46]. Other papers introduced techniques such as modulating an excitation signal into a chirp [35], on-off keying to reduce interference and reduce spectrum occupation [15] and parallel decoding and long-range support [24].

Compared to existing backscattering methods [6, 15, 22, 25, 26, 35, 51], *SideComm has the following advantages*. First, backscatter systems typically require to be placed close (within a meter) to their active transmitter or receiver device in most scenarios such

as NLOS [11]. Moreover, their tag typically requires a high-gain antenna. In contrast, SideComm does not require any antenna nor being placed close to the receiver.

7 CONCLUSIONS

SideComm is a system that utilizes electromagnetic side-channels to enable low-power and >10m communication range for resource-limited Internet-of-Things devices, without any additional hardware or need for a complex memory system. While side-channels have traditionally been used for adversarial purposes, this paper uses side-channels for the beneficial purpose of creating connectivity for IoT devices. The key was to leverage software activities to create LoRa-like packets. We demonstrated SideComm's range via SNR measurements at extended distances, as well as SideComm's resistance to attenuation via non-line-of-sight measurements. Finally, we demonstrated SideComm's comparatively lower power consumption compared to existing transmitter modules and technologies. SideComm provides a promising method of communication for many smart environments by turning a microprocessor into a radio that is simple, low-power, and flexible.

ACKNOWLEDGMENTS

We thank our shepherd for their guidance. This work has been supported, in part, by NSF grant 2211301 and IARPA grant PO-IARPA 2021-21062400004. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF and IARPA.

REFERENCES

[1] Ali Abedi, Omid Abari, and Tim Brecht. 2019. Wi-le: Can wifi replace bluetooth?. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*. 117–124.
[2] Ali Abedi, Mohammad Hossein Mazaheri, Omid Abari, and Tim Brecht. 2018. Witag: Rethinking backscatter communication for wifi networks. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*. 148–154.
[3] Orion Afisiadis, Andreas Burg, and Alexios Balatsoukas-Stimming. 2020. Coded LoRa frame error rate analysis. In *Icc 2020-2020 Ieee International Conference On Communications (Icc)*. IEEE, 1–6.
[4] Indoor TV Antenna. [n. d.]. https://www.amazon.com/dp/B01FUB4ZG8?psc=1&ref=ppx_yo2ov_dt_b_product_details. Accessed: 2022-07.
[5] USPR B205mini-i. [n. d.]. https://www.ettus.com/all-products/usrp-b205mini-i/. Accessed: 2022-07.
[6] Dinesh Bharadia, Kiran Raj Joshi, Manikanta Kotaru, and Sachin Katti. 2015. Backfi: High throughput wifi backscatter. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 283–296.
[7] Robert Callan, Alenka Zajić, and Milos Prvulovic. 2015. FASE: Finding amplitude-modulated side-channel emanations. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 592–603.
[8] G. Camurati and A. Francillon. 2022. Noise-SDR: Arbitrary Modulation of Electromagnetic Noise from Unprivileged Software and Its Impact on Emission Security. In *2022 2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 294–311. https://doi.org/10.1109/SP46214.2022.00018
[9] Giovanni Camurati, Aurélien Francillon, and François-Xavier Standaert. [n. d.]. Understanding screaming channels: From a detailed analysis to improved attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* ([n. d.]).
[10] TI CC2650 BLE Chip. [n. d.]. https://www.ti.com/product/CC2650/. Accessed: 2022-07.
[11] Farzan Dehbashi, Ali Abedi, Tim Brecht, and Omid Abari. 2021. Verification: can wifi backscatter replace RFID?. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 97–107.
[12] Amalinda Gamage, Jansen Christian Liando, Chaojie Gu, Rui Tan, and Mo Li. 2020. Lmac: Efficient carrier-sense multiple access for lora. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*.
[13] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. 2015. Stealing keys from PCs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *International workshop on cryptographic hardware and embedded systems*. Springer, 207–228.
[14] Reza Ghanaatian, Orion Afisiadis, Matthieu Cotting, and Andreas Burg. 2019. LoRa digital receiver analysis and implementation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.
[15] Xiuzhen Guo, Longfei Shangguan, Yuan He, Jia Zhang, Haotian Jiang, Awais Ahmad Siddiqi, and Yunhao Liu. 2020. Aloba: rethinking ON-OFF keying modulation for ambient LoRa backscatter. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 192–204.
[16] Mordechai Guri. 2020. AIR-FI: Generating covert wi-fi signals from air-gapped computers. *arXiv preprint arXiv:2012.06884* (2020).
[17] Mordechai Guri, Assaf Kachlon, Ofer Hasson, Gabi Kedma, Yisroel Mirsky, and Yuval Elovici. 2015. {GSMem}: Data Exfiltration from {Air-Gapped} Computers over {GSM} Frequencies. In *24th USENIX Security Symposium*. 849–864.
[18] Mordechai Guri, Gabi Kedma, Assaf Kachlon, and Yuval Elovici. 2014. AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies. In *2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE)*. IEEE, 58–67.
[19] Yi Han, Matthew Chan, Zahra Aref, Nils Ole Tippenhauer, and Saman Zonouz. 2022. Hiding in Plain Sight? On the Efficacy of Power Side Channel-Based Control Flow Monitoring. In *Proceedings of the USENIX Security Symposium*.
[20] Yi Han, Sriharsha Etigowni, Hua Liu, Saman Zonouz, and Athina Petropulu. 2017. Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1095–1108.
[21] Jiaji He, Yiqiang Zhao, Xiaolong Guo, and Yier Jin. 2017. Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 10 (2017).
[22] Mehrdad Hessar, Ali Najafi, and Shyamnath Gollakota. 2019. {NetScatter}: Enabling {Large-Scale} Backscatter Networks. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*. 271–284.
[23] Adafruit INA219. [n. d.]. https://www.adafruit.com/product/904/. Accessed: 2022-07.
[24] Jinyan Jiang, Zhenqiang Xu, Fan Dang, and Jiliang Wang. 2021. Long-range ambient LoRa backscatter with parallel decoding. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 684–696.
[25] 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*. 607–618.
[26] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. 2016. Passive {Wi-Fi}: Bringing Low Power to {Wi-Fi} Transmissions. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 151–164.
[27] Chenning Li and Zhichao Cao. 2022. Lora networking techniques for large-scale and long-term iot: A down-to-top survey. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–36.
[28] Chenning Li, Hanqing Guo, Shuai Tong, Xiao Zeng, Zhichao Cao, Mi Zhang, Qiben Yan, Li Xiao, Jiliang Wang, and Yunhao Liu. 2021. NELoRa: Towards Ultra-low SNR LoRa Communication with Neural-enhanced Demodulation. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*.
[29] Yinghui Li, Jing Yang, and Jiliang Wang. 2020. Dylora: Towards energy efficient dynamic lora transmission control. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2312–2320.
[30] Zhijun Li and Yongrui Chen. 2020. BLE2LoRa: cross-technology communication from bluetooth to LoRa via chirp emulation. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE.
[31] Jun Liu, Jiayao Gao, Sanjay Jha, and Wen Hu. 2021. Seirios: leveraging multiple channels for LoRaWAN indoor and outdoor localization. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 656–669.
[32] Ruizhe Long, Ying-Chang Liang, Huayan Guo, Gang Yang, and Rui Zhang. 2019. Symbiotic radio: A new communication paradigm for passive Internet of Things. *IEEE Internet of Things Journal* 7, 2 (2019), 1350–1363.
[33] Alexandre Marquet, Nicolas Montavont, and Georgios Z Papadopoulos. 2020. Towards an SDR implementation of LoRa: Reverse-engineering, demodulation strategies and assessment over Rayleigh channel. *Computer Communications* 153 (2020), 595–605.
[34] TI MSP430. [n. d.]. https://www.ti.com/tool/MSP-EXP430G2ET/. Accessed: 2022-07.
[35] 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*. 147–160.
[36] Yaman Sangar and Bhuvana Krishnaswamy. 2020. WiChronos: energy-efficient modulation for long-range, large-scale wireless networks. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–14.
[37] Seun Sangodoyin, Frank T Werner, Baki B Yilmaz, Chia-Lin Cheng, Elvan M Ugurlu, Nader Sehatbakhsh, Milos Prvulović, and Alenka Zajic. 2020. Side-channel propagation measurements and modeling for hardware security in iot devices. *IEEE Transactions on Antennas and Propagation* 69, 6 (2020), 3470–3484.
[38] Nader Sehatbakhsh, Alireza Nazari, Monjur Alam, Frank Werner, Yuanda Zhu, Alenka Zajic, and Milos Prvulovic. 2019. REMOTE: Robust external malware

detection framework by using electromagnetic signals. *IEEE Trans. Comput.* 69, 3 (2019), 312–326.

[39] Nader Sehatbakhsh, Alireza Nazari, Alenka Zajic, and Milos Prvulovic. 2016. Spectral profiling: Observer-effect-free profiling by monitoring EM emanations. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 1–11.

[40] Nader Sehatbakhsh, Baki Berkay Yilmaz, Alenka Zajic, and Milos Prvulovic. 2020. A new side-channel vulnerability on modern computers by exploiting electromagnetic emanations from the power management unit. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.

[41] Cheng Shen, Tian Liu, Jun Huang, and Rui Tan. 2021. When LoRa meets EMR: Electromagnetic covert channels can be super resilient. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1304–1317.

[42] Pang Shuai, Zhao Yonghui, and Ouyang Jineng. 2021. An Intelligent Wireless Networking and Application Layer Protocol Design Based on LoRa. In *2021 IEEE 4th International Conference on Electronics Technology (ICET)*. IEEE, 1099–1104.

[43] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, and Johanna Nieminen. 2012. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *2012 IEEE wireless communications and networking conference workshops (WCNCW)*. IEEE, 232–237.

[44] STM32WL55. [n. d.]. https://www.st.com/en/microcontrollers-microprocessors/stm32wl55jc.html. Accessed: 2022-07.

[45] Semtech SX1262. [n. d.]. https://www.semtech.com/products/wireless-rf/lora-core/sx1262mb2cas/. Accessed: 2022-07.

[46] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R Smith, and Shyamnath Gollakota. 2017. Lora backscatter: Enabling the vision of ubiquitous connectivity. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 1, 3 (2017), 1–24.

[47] Joachim Tapparel, Orion Afisiadis, Paul Mayoraz, Alexios Balatsoukas-Stimming, and Andreas Burg. 2020. An open-source LoRa physical layer prototype on GNU radio. In *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 1–5.

[48] Shuai Tong, Zilin Shen, Yunhao Liu, and Jiliang Wang. 2021. Combating link dynamics for reliable lora connection in urban settings. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 642–655.

[49] Shuai Tong, Jiliang Wang, and Yunhao Liu. 2020. Combating packet collisions using non-stationary signal scaling in LPWANs. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 234–246.

[50] Arduino Uno. [n. d.]. https://docs.arduino.cc/hardware/uno-rev3/. Accessed: 2022-07.

[51] Ambuj Varshney, Oliver Harms, Carlos Pérez-Penichet, Christian Rohner, Frederik Hermans, and Thiemo Voigt. 2017. Lorea: A backscatter architecture that achieves a long communication range. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–14.

[52] 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 Conference of the ACM Special Interest Group on Data Communication*.

[53] Binbin Xie and Jie Xiong. 2020. Combating interference for long range LoRa sensing. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 69–81.

[54] Yuval Yarom and Katrina Falkner. 2014. {FLUSH+ RELOAD}: A High Resolution, Low Noise, L3 Cache {Side-Channel} Attack. In *23rd USENIX security symposium (USENIX security 14)*. 719–732.

[55] Zihao Zhan, Zhenkai Zhang, and Xenofon Koutsoukos. 2020. Bitjabber: The world's fastest electromagnetic covert channel. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 35–45.

[56] Zhenkai Zhang, Zihao Zhan, Daniel Balasubramanian, Bo Li, Peter Volgyesi, and Xenofon Koutsoukos. 2020. Leveraging EM side-channel information to detect rowhammer attacks. In *2020 IEEE Symposium on Security and Privacy (SP)*.

[57] Mark Zhao and G Edward Suh. 2018. FPGA-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 229–244.