An Advanced GNU Radio Receiver of IEEE 802.15.4 OQPSK Physical Layer

Evan Faulkner, Zelin Yun, Shengli Zhou, Zhijie Shi, Song Han, and Georgios B. Giannakis

***Abstract*—In this paper, we present an advanced coherent receiver for the IEEE 802.15.4 offset-quadrature-phase-shift-keying (OQPSK) physical layer and provide an open-source implementation in the GNU Radio framework. Simulation and field test results show that the proposed receiver achieves about 11 dB power gain over an existing GNU Radio receiver, which treats OQPSK as minimum-shift-keying (MSK) for low-complexity processing. While suitable modules can be added to the MSK-based receiver for performance enhancement, the proposed receiver still maintains a 6 dB power gain. The proposed coherent receiver is attractive for IoT applications where a powerful software-defined-radio (SDR) based gateway is deployed to interact with various sensors.**

***Index Terms*—IoT, IEEE 802.15.4, OQPSK, SDR, GNU-Radio**

I. Introduction

THE IEEE 802.15.4 standard was developed in 2003 and has since undergone multiple rounds of revisions [1], [2]. The standard deals with physical layer and medium access control techniques for numerous low-rate wireless local area networks such as ZigBee, 6LoWPAN, and WirelessHART [3]–[7], and will remain popular for emerging IoT applications [3].

This paper focuses on the physical layer that relies on Offset Quadrature Phase Shift Keying (OQPSK). At the transmitter, each group of four bits in the data packet is mapped to a sequence of 32 chips, which corresponds to a direct-sequence spread spectrum (DSSS) operation. The chip sequence is divided into I and Q branches and passes through a half-sine pulse shaper. The waveform on the Q branch is delayed by one chip period, and hence an OQPSK waveform is generated. The overall process yields a carefully designed 16-ary modulation scheme consisting of 16 nearly-orthogonal waveforms. On the other hand, OQPSK can be viewed as minimum-shift-keying (MSK) modulation over a transformed chip sequence [8]. For clarity, we next discuss existing IEEE 802.15.4 receiver designs depending on whether the modulation is treated as OQPSK or MSK.

A. OQPSK-based Receiver

A complete receiver includes modules for carrier frequency synchronization, chip/symbol timing, and data detection, although the orders might change depending on a particular implementation. While simulation-based studies tend to focus on one or two key modules, a fully-functional receiver requires all modules to be implemented. We categorize existing references according to the modules they address:

- *Data detection.* In the presence of additive white Gaussian noise (AWGN), tight union bounds on the probability of error for both coherent and noncoherent maximum likelihood (ML) detectors with known signal parameters are derived in [9]. A simple detector based on a double-correlation operation is developed in [10] that nearly reaches the performance of noncoherent ML detector in the presence of a large carrier frequency offset (CFO). The noncoherent demodulator in [11] replaces the complex correlation metrics in the coherent ML receiver by their magnitudes, which achieves robustness against phase mismatch and frequency offset.
- *Synchronization.* A carrier synchronization algorithm is proposed in [12] by measuring the slope of phase drifts on the complex baseband samples corresponding to the synchronization header of the data packet. In [13], carrier synchronization is accomplished by the Costa's loop and symbol synchronization is pursued via the early-late-gate timing recovery method.
- *Full-receiver implementation.* A digital baseband transceiver is implemented in [14] using FPGA and CMOS technologies, where a noncoherent demodulator is adopted. A coherent receiver is implemented in [15] using MATLAB based on data samples from a software-defined-radio (SDR) device. Similarly, an extensive evaluation of a coherent receiver is carried out in [16] using MATLAB. Recently, a coherent OQPSK receiver with a decision-directed residual phase noise compensation procedure has been implemented in [17] under a dual-mode architecture, where the receiver could switch into low-complexity MSK-based processing depending on channel conditions or performance requirements. The dual-mode receiver is evaluated in [17] along with FPGA prototyping and ASIC implementation.

B. MSK-based Receiver

The receiver treats the OQPSK waveform as an analog MSK waveform and uses a phase differential operation to obtain an estimate of the instantaneous frequency of the received signal. Timing recovery is then applied to achieve chip level synchronization, followed by a despreading operation for data detection. This receiver bypasses the need for carrier frequency synchronization and timing recovery is operated on real, instead of complex, baseband signals.

Evan Faulkner and Shengli Zhou are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, 06250.

Zelin Yun, Zhijie Shi, and Song Han are with the Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, 06250.

Georgios B. Giannakis is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, 55455.

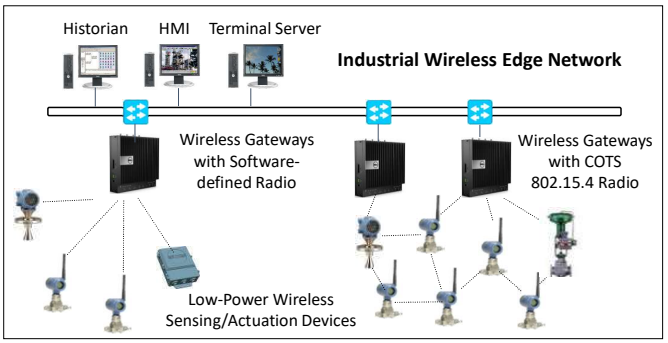

Fig. 1. Various sensors communicate with powerful SDR-based gateways.

The MSK-based receiver tailored for IEEE 802.15.4 was first presented in [18]. A GNU radio implementation was provided in [19] and later refined in [20], [21]. GNU radio implementations are open source and hence valuable to users and developers. Indeed, the implementation in [20], [21] has been popular in the community and has led to numerous follow-on works, e.g., [22]–[26]. Specifically, dynamic spectrum access has been explored based on 802.15.4 networks in [22]. A new synchronization scheme without preamble symbols has been developed in [23]. A physical layer SDR testbed is implemented in [24] to allow rapid prototyping of wireless sensor networks. An efficient error packet recovery is presented in [25] based on error characteristic of adjacent data symbols. Using the commodity SDR hardware and an open-source software testbed, characterization of ZigBee devices from different manufactures has been demonstrated in [26].

C. Contributions of This work

In this work, we introduce a novel coherent receiver for the IEEE 802.15.4 OQPSK physical layer and implement it in the GNU radio framework. The proposed receiver has three stages. In the first stage, packet detection is carried out in the presence of an unknown CFO based on the specific preamble in IEEE 802.15.4. In the second stage, CFO is estimated and symbol timing is refined. In the third stage, a linear decision-directed equalizer is adopted for symbol detection, which effectively tracks residual frequency variations and equalizes the multipath channel.

The proposed receiver is distinct from all the OQPSK receivers discussed in Section I-A. In particular, no existing receivers have adopted the presented triggering mechanism in the presence of unknown CFO and the linear equalizer approach to track the residual frequency change and mitigate the channel time-dispersion. In contrast to the MSK-based GNU Radio receivers discussed in Section I-B, this work presents the first GNU Radio implementation of an OQPSK-based receiver. Simulation results show that the proposed receiver achieves a 11 dB performance improvement over the existing GNU radio receiver [20], [21], and maintains a 6 dB gain after we apply modifications to the MSK-based receiver to enhance its performance. A field test in an indoor environment verifies that the proposed receiver maintains large performance gains over both the existing GNU radio receiver and the hardware CC2652 receiver from Texas Instruments.

The excellent performance of our novel GNU Radio receiver is achieved at the expense of receiver complexity. The proposed receiver would not be preferred for a low-cost or low-power receiver. However, it is appealing for application scenarios as shown in Fig. 1, where wireless nodes (including both sensors and actuators) must operate with ultra-low power consumption and may come from different manufacturers while the gateways can be more powerful and can be implemented via SDR [27]–[29]. Compared with existing gateways deployed in the field with low-cost receivers, the proposed receiver can significantly increase the communication range for the uplink transmission from the wireless nodes to the gateways. This increased communication range will simplify the network topology design, require a smaller number of relay nodes deployed in the field, and reduce the deployment and maintenance costs of the system. For the transmissions from the gateway to the wireless nodes, the gateway is often connected to the backbone network and not energy constrained and can therefore offer a high-power transmission to the low-power receiver nodes.

The rest of this paper is organized as follows. Section II reviews the transmitter design, and Section III develops a coherent receiver. Section IV presents the GNU radio implementation, with its simulated performance demonstrated in Section V and field test results collected in Section VI. Section VII presents the conclusions.

Notation: Boldface lower case letters stand for row vectors and upper case letters stand for matrices. Symbol $|\cdot|$ stands for the absolute value, $\|\cdot\|$ for the 2-norm of a vector, and $(\cdot)^H$ for the Hermitian transpose of a vector or matrix. For two vectors $\mathbf{a}$ and $\mathbf{b}$, the correlation coefficient is defined as

$$< \mathbf{a}, \mathbf{b} > = \frac{|\mathbf{a}\mathbf{b}^H|}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}. \tag{1}$$

II. IEEE 802.15.4 OQPSK PHY

Consider the popular OQPSK physical layer in the IEEE 802.15.4 standard. The packet structure is shown in Fig. 2. Each packet consists of a preamble, a header, and a payload [1]. The preamble contains 4 bytes of zeros followed by a start of frame delimiter, 0xA7. A one-byte packet header gives the number of bytes contained in the payload of the packet. One bit in the header is reserved so that the maximum packet payload length is 127 bytes.

All bytes in the packet are represented with the most significant bit on the right. Each byte is split into two 4-bit symbols, and each 4-bit symbol is mapped to one of the 16 nearly orthogonal 32-chip sequences given in Fig. I. Even-indexed chips are modulated onto the data sequence I_n on the in-phase carrier and odd-indexed chips are modulated to the data sequence Q_n on the quadrature-phase carrier with the mapping $0 \rightarrow -1$ and $1 \rightarrow 1$. Let T_c denote the chip period and $g(t)$ denote the half-sine pulse

$$g(t) = \sin\left(\frac{\pi t}{2T_c}\right), \ 0 \le t \le 2T_c. \tag{2}$$

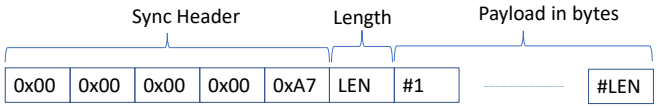

Fig. 2. Packet structure

TABLE I
THE 32 CHIP PSEUDORANDOM QUASI-ORTHOGONAL SEQUENCES FOR THE OQPSK PHYSICAL LAYER [1].

Data symbol	Chip values $\{c_0 c_1 \cdots c_{30} c_{31}\}$
0	11011001110000110101001000101110
1	11101101100111000011010100100010
2	00101110110100111000011010100010
3	00100010111011011001110000110101
4	01010010001011101101100111000011
5	00110101001000101110110110011100
6	11000011010100100010111011011001
7	10011100001101010010001011101101
8	10001100100101100000011101111011
9	10111000110010010110000001110111
10	01111011100011001001011000000111
11	01110111011100011001001011000000
12	00000111011110111000110010010110
13	01100000011101111011100011001001
14	10010110000001110111101110001100
15	11001001011000000111011110111000

The baseband OQPSK signal is

$$s(t) = \sum_n I_n g(t - 2nT_{\rm c}) + j \sum_n Q_n g(t - 2nT_{\rm c} - T_{\rm c}), \quad (3)$$

where the quadrature phase signal is delayed by one chip period.

In an SDR implementation, the data samples from $s(t)$ are passed from the computer to the SDR device. The baseband waveforms are converted to passband via the SDR unit as

$$s^{\rm pb}(t) = {\rm Re}\{s(t)e^{j2\pi f_{\rm c,tx} t}\}, \quad (4)$$

where $f_{\rm c,tx}$ is the carrier frequency at the transmitter. The chip duration is $T_{\rm c} = 0.5\ \mu s$, from which we can infer the symbol rate $(1/T_{\rm c})/32 = 62.5$ kilo-symbols/s and the data rate of the OQPSK PHY is $62.5 \times 4 = 250$ kb/s.

III. COHERENT RECEIVER DESIGN

The received passband waveform $x^{\rm pb}(t)$ is converted to the baseband as

$$x(t) = {\rm LPF}\{x^{\rm pb}(t)e^{-j2\pi f_{\rm c,rx} t}\}, \quad (5)$$

where $f_{\rm c,rx}$ is the carrier frequency at the receiver. Due to imperfect oscillators, there exists a non-negligible carrier frequency offset (CFO)

$$\epsilon = f_{\rm c,tx} - f_{\rm c,rx}. \quad (6)$$

If multipath dispersion is explicitly modelled, the channel input-output at the baseband can be represented as:

$$x(t) = e^{j2\pi\epsilon t} \sum_l h_l s(t - \tau_l) + w(t), \quad (7)$$

where τ_l is the delay and h_l is the amplitude of the lth path, and $w(t)$ is additive white Gaussian noise (AWGN).

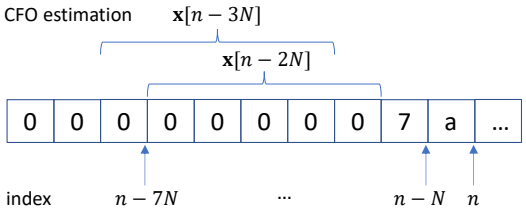

Fig. 3. Illustration of the receiver template and index.

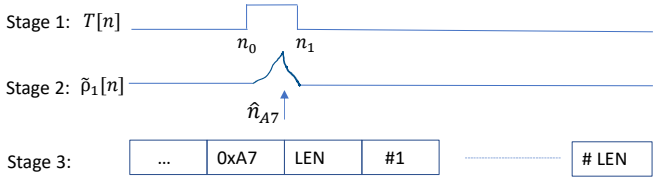

Fig. 4. Sample index used in three stages of receiver processing.

The sampling rate $f_{\rm s}$ is set as 4 MHz in this paper. The sampling interval is $t_{\rm s} = 1/f_{\rm s}$, and the sampled sequence is

$$x[n] = x(t)|_{t=nt_{\rm s}}. \quad (8)$$

These samples are passed from an SDR unit to the computer for further processing.

We next present a coherent receiver to decode the data from the discrete samples. Reception and decoding of a transmission require detection of the packet, carrier frequency offset compensation, timing synchronization, and compensation for channel effects. For convenience, let us define the templates corresponding to 0, 7, and a symbols as

$$\mathbf{s}_0, \mathbf{s}_7, \mathbf{s}_a. \quad (9)$$

Further, define two templates corresponding to the 0x00 and 0xA7 bytes as:

$$\mathbf{d}_{00} = \big[\mathbf{s}_0, \mathbf{s}_0\big], \quad \mathbf{d}_{\rm A7} = \big[\mathbf{s}_7, \mathbf{s}_a\big]. \quad (10)$$

Note that $\mathbf{s}_7$ is in front of $\mathbf{s}_a$ due to the format of the most significant bits on the right. With a sampling rate of 4 MHz, templates $\mathbf{s}_0$, $\mathbf{s}_7$, $\mathbf{s}_a$ have $N = 64$ samples while templates $\mathbf{d}_{00}$ and $\mathbf{d}_{\rm A7}$ have $M = 2N = 128$ samples.

The following subsections describe each of the three stages of our coherent receiver, with useful illustrations given in Fig. 3 and Fig. 4.

A. First Stage: Triggering

To trigger the receiver processing in the presence of unknown CFO, we adopt a set of parallel branches where each branch has been compensated by a fixed CFO value. On the ith branch, the CFO compensation is applied to obtain

$$y_i[n] = x[n]e^{-j2\pi\epsilon_i n t_{\rm s}}, \quad (11)$$

where ϵ_i is the CFO value on the ith branch. At each sample index n, define a length-N vector

$$\mathbf{y}_i[n] = \big[y_i[n-N+1], \ldots, y_i[n]\big]. \quad (12)$$

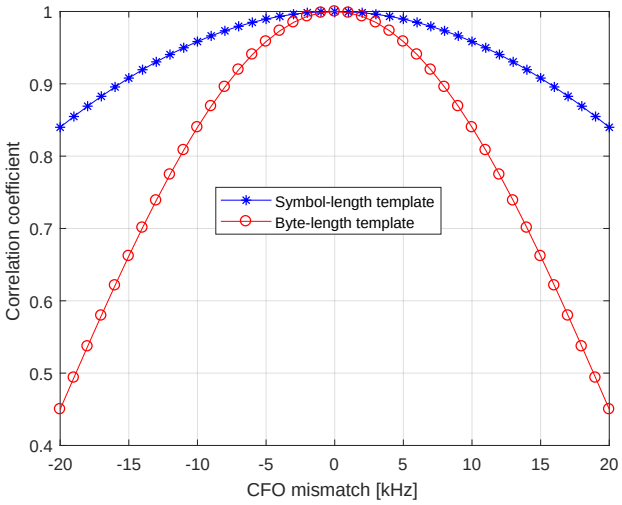

Fig. 5. Correlation height in the presence of CFO mismatch.

For each n, three correlation outputs are computed as

$$\rho_0[n] = < \mathbf{y}_i[n], \mathbf{s}_0 >, \tag{13}$$

$$\rho_7[n] = < \mathbf{y}_i[n], \mathbf{s}_7 >, \tag{14}$$

$$\rho_a[n] = < \mathbf{y}_i[n], \mathbf{s}_a > . \tag{15}$$

We choose correlation at the symbol level of length N rather than at the byte level of length $M = 2N$ because the CFO mismatch leads to a reduced correlation peak. As shown in Fig. 5, the symbol level correlation is more robust to CFO mismatch than the byte level correlation. A 16-kHz CFO mismatch reduces the correlation peak from 1 to 0.9 using the symbol level correlation.

Following the pattern illustrated in Fig. 3, a trigger signal on the ith branch is defined as

$$T_i[n] = \prod_{i=2}^{7} \big(\rho_0[n-iN] \geq \Gamma_1\big)\big(\rho_7[n-N] \geq \Gamma_1\big)\big(\rho_a[n] \geq \Gamma_1\big), \tag{16}$$

where Γ_1 is a threshold to be set. Combining the output from P branches, the overall trigger signal as:

$$T[n] = T_1[n] \,\big|\, T_2[n] \,\big|\, \cdots \big|\, T_P[n], \tag{17}$$

where $|$ stands for the OR operation. When $T[n] = 0$, there is no follow-on processing. The second stage processing is activated during each segment where $T[n] = 1$.

B. Second Stage: Synchronization and Packet Detection

Second-stage processing is performed within each data segment where $T[n] = 1$. For one segment, denote the starting index as n_0 and the ending index as n_1, as shown in Fig. 4. The following tasks are carried out sequentially.

1) CFO estimation: CFO estimation is done using the samples indicated in Fig. 3 by leveraging the last three 0x00 bytes in the preamble, where the first 0x00 byte is not used in order to avoid problems caused by the transient effects of initiating a transmission in the hardware.

The overall CFO has fractional and integer CFO components as

$$\epsilon = \varepsilon_{\rm f} + \nu\Delta, \tag{18}$$

where ν is an integer, Δ is a constant, and $\epsilon_{\rm f}$ is the fractional CFO in the range of $(-\Delta/2, \Delta/2)$. Define a length-$5N$ vector $\mathbf{x}[n]$ at time n as

$$\mathbf{x}[n] = \big[x[n-5N+1], \ldots, x[n]\big]. \tag{19}$$

We estimate the fractional CFO via

$$\hat{\epsilon}_{\rm f} = \frac{\angle\big(\mathbf{x}[n_0-2N]\mathbf{x}^H[n_0-3N]\big)}{2\pi N t_{\rm s}}, \tag{20}$$

where the operation $\angle(\cdot)$ evaluates the angle of a complex number with units in radians. Since the output of $\angle(\cdot)$ is limited to $(-\pi, \pi)$, and $Nt_{\rm s} = 16\ \mu s$, the fractional CFO $\hat{\epsilon}_{\rm f}$ is limited in the range of $(-31.25, 31.25)$ kHz with the constant $\Delta = 62.5$ kHz.

We resolve the integer ν from the set $\{-I, \ldots, I\}$ with $2I+1$ candidate values. For each ν, define a CFO-compensated sequence

$$\tilde{y}_\nu[n] = x[n]e^{-j2\pi(\hat{\epsilon}_{\rm f}+\nu\Delta)nt_{\rm s}}. \tag{21}$$

Further, collect the samples into a length-M vector as

$$\tilde{\mathbf{y}}_\nu[n_0] = \big[\tilde{y}_\nu[n_0-M+1], \ldots, \tilde{y}_\nu[n_0]\big]. \tag{22}$$

The integer CFO estimate is found by

$$\hat{\nu} = \arg \max_{\nu=-I,\ldots,I} < \tilde{\mathbf{y}}_\nu[n_0], \mathbf{d}_{\rm A7} > . \tag{23}$$

The overall CFO is estimated by

$$\hat{\epsilon} = \hat{\varepsilon}_{\rm f} + \hat{\nu}\Delta. \tag{24}$$

The estimate $\hat{\epsilon}$ falls in the range of

$$(-31.25 - 62.5I, +31.25 + 62.5I) \text{ kHz}. \tag{25}$$

We set $I = 2$ in the receiver implementation.

2) Fine timing: Fine timing is carried out based on the CFO compensated samples

$$\tilde{y}[n] = e^{-j2\pi\hat{\epsilon}nt_{\rm s}}x[n]. \tag{26}$$

At each index n, define a vector

$$\tilde{\mathbf{y}}[n] = [\tilde{y}[n-M+1], \ldots, \tilde{y}[n]]. \tag{27}$$

A correlation with the 0xA7 template leads to

$$\tilde{\rho}_1[n] = < \tilde{\mathbf{y}}[n], \mathbf{d}_{\rm A7} > . \tag{28}$$

During each segment where $T[n] = 1$, the best sample index for the last sample of the 0xA7 byte is estimated as:

$$\hat{n}_{\rm A7} = \arg \max_{n_0 \leq n \leq n_1} \tilde{\rho}_1[n]. \tag{29}$$

3) Packet detection: To reduce the probability of false alarms, we further verify that the bytes preceding 0xA7 are indeed 0x00. Define correlation outputs as

$$\tilde{\rho}_0[n] = < \tilde{\mathbf{y}}[n], \mathbf{d}_{00} >, \tag{30}$$

where $\tilde{\mathbf{y}}[n]$ is defined in (27). The packet detection variable is determined as:

$$D = \big(\tilde{\rho}_1[\hat{n}_{\rm A7}] \geq \Gamma_2\big) \prod_{i=2}^{6} \big(\tilde{\rho}_0[\hat{n}_{\rm A7} - iN] \geq \Gamma_2\big), \tag{31}$$

where Γ_2 is a pre-defined threshold.

At each instance when $D = 1$, a packet arrival is declared, and the third-stage processing is activated. The timing offset $\hat{n}_{\rm A7}$ and the CFO estimate $\hat{\epsilon}$ are passed to the third stage.

4) Link Quality Indicator: Per detected packet, the received signal strength indicator (RSSI) is the energy of the received samples corresponding to the three 0x00 bytes as:

$$E = \frac{1}{6N} \sum_{n=\hat{n}_{\rm A7}-8N+1}^{\hat{n}_{\rm A7}-2N} |x[n]|^2. \tag{32}$$

The link quality indicator (LQI) can be evaluated by an estimate on the signal to noise ratio (SNR) as follows. Define

$$\zeta = < \mathbf{x}[\hat{n}_{\rm A7} - 3N], \mathbf{x}[\hat{n}_{\rm A7} - 2N] >, \tag{33}$$

where $\mathbf{x}[n]$ is defined in (19). The expected value of the correlation is $\zeta = \text{SNR}/(1 + \text{SNR})$, and hence the SNR can be estimated as:

$$\text{SNR} = \frac{\zeta}{1 - \zeta}. \tag{34}$$

C. Third Stage: Data Detection

With the correct timing and the estimated CFO, we obtain the compensated data sequence as

$$r[n] = e^{-j2\pi\hat{\epsilon}nt_{\rm s}} x[n + \hat{n}_{\rm A7}], \quad n = 1, 2, ... \tag{35}$$

We adopt a length-K linear equalizer for channel equalization as detailed in [30]. Corresponding to the lth symbol to be detected, the equalizer coefficients are collected into a vector of $K = K_1 + K_2 + 1$ taps as

$$\mathbf{f}[l] = \Big[f[l; -K_1], ..., f[l; 0], ...f[l; K_2]\Big]. \tag{36}$$

We will rely on decision-directed equalization. Let $\hat{\mathbf{f}}[l-1]$ be the equalizer computed based on the $(l-1)$-th symbol. For the l-th symbol, the equalizer output is:

$$\hat{\mathbf{z}}[l] = \hat{\mathbf{f}}[l-1]\mathbf{R}[l], \tag{37}$$

where

$$\mathbf{z}[l] = [z[lN - N + 1], \cdots, z[lN]], \tag{38}$$

$$\mathbf{R}[l] = \begin{bmatrix} r[lN - N + 1 - K_1] & \dots & r[lN - K_1] \\ \vdots & & \vdots \\ r[lN - N + 1] & \dots & r[lN] \\ \vdots & & \vdots \\ r[lN - N + 1 + K_2] & \dots & r[lN + K_2] \end{bmatrix}. \tag{39}$$

The equalized sequence is used to detect the symbol through the maximum-correlation criterion as:

$$\hat{\mathbf{s}}[l] = \arg\max_{\mathbf{s}} \text{Re}\{\hat{\mathbf{z}}[l]\mathbf{s}^H\}, \tag{40}$$

where $\mathbf{s}$ is the data sequence corresponding to the 16 possible chip sequences in Table I.

Assuming that the decoded symbol $\hat{\mathbf{s}}[l]$ is correct, the equalizer at the lth symbol is updated as:

$$\hat{\mathbf{f}}[l] = \hat{\mathbf{s}}[l]\mathbf{R}^H[l]([\mathbf{R}[l]\mathbf{R}^H[l])^{-1}. \tag{41}$$

For initialization, $\hat{\mathbf{f}}[0]$ is computed based on the 0xA7 byte in the preamble. The equalizer update and the data detection order is:

$$\hat{\mathbf{f}}[0] \to \hat{\mathbf{s}}[1] \to \hat{\mathbf{f}}[1] \to \hat{\mathbf{s}}[2] \to \cdots \tag{42}$$

Typically, the decision directed approach is susceptible to error propagation where one incorrect decision will impact the detection of later symbols. However, a single symbol error means that the whole packet is in error, and hence the error propagation does not negatively affect the packet delivery ratio.

A basic symbol-by-symbol update is presented here for convenience. Alternatively, one can update the equalizer every several symbols. For noise suppression, one can update the equalizer by stacking several symbols together. One convenient variation is to update the equalizer byte-by-byte, where each byte contains two symbols.

IV. GNU Radio Implementation

The GNU Radio IEEE 802.15.4 testbed in [20], [21] implements the communication stack from the physical layer up to the network layer, and allows the application layer to be attached easily. This implementation is included in the GNU Radio 3.8 release and is popular in the GNU Radio community. This paper only focuses on the receiver portion of the OQPSK physical layer, and aims to improve the error performance.

A. Enhancing the MSK-based receiver [19]–[21]

Fig. 6 shows the flowgraph of the MSK-based receiver [20], [21] in gnuradio-companion (GRC). There are four key modules.

- The `Quadrature Demod` module takes the phase difference of consecutive samples $\angle(x^*[n-1]x[n])$, which is an estimate of the instantaneous frequency of the received signal.
- The `Single Pole IIR Filter` module tracks the DC bias from the frequency estimates due to the carrier frequency shift. The DC bias is subtracted from the frequency estimates.
- The `Clock Recovery MM` module implements the Müller and Mueller timing recovery algorithm [31] and outputs a recovered chip sequence.
- The custom-made `Packet Sink` module performs frame synchronization and data detection, and outputs the decoded packet.

We recommend to add two modules to improve the performance of the MSK-based receiver.

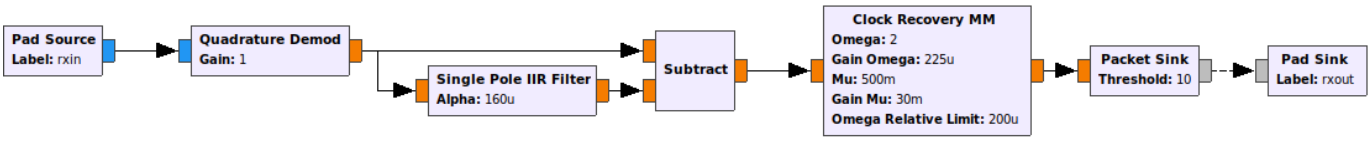

Fig. 6. The current MSK-based receiver in [21].

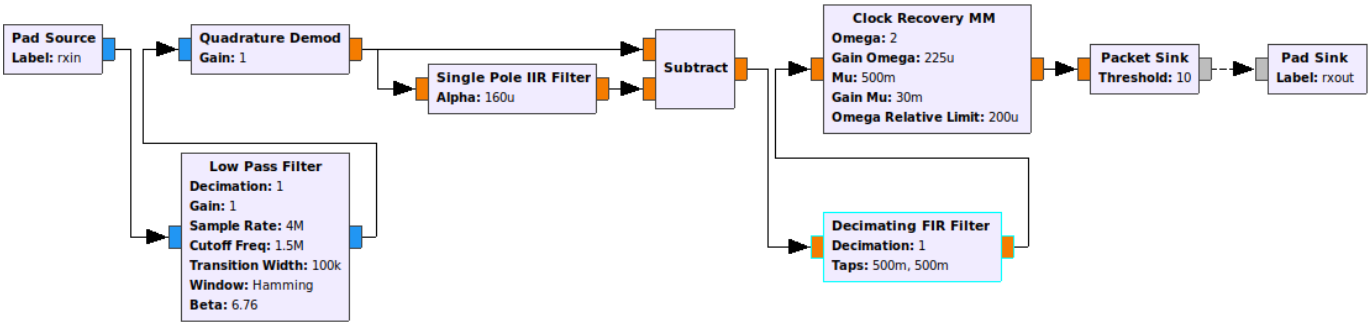

Fig. 7. The proposed enhancement for the MSK-based receiver.

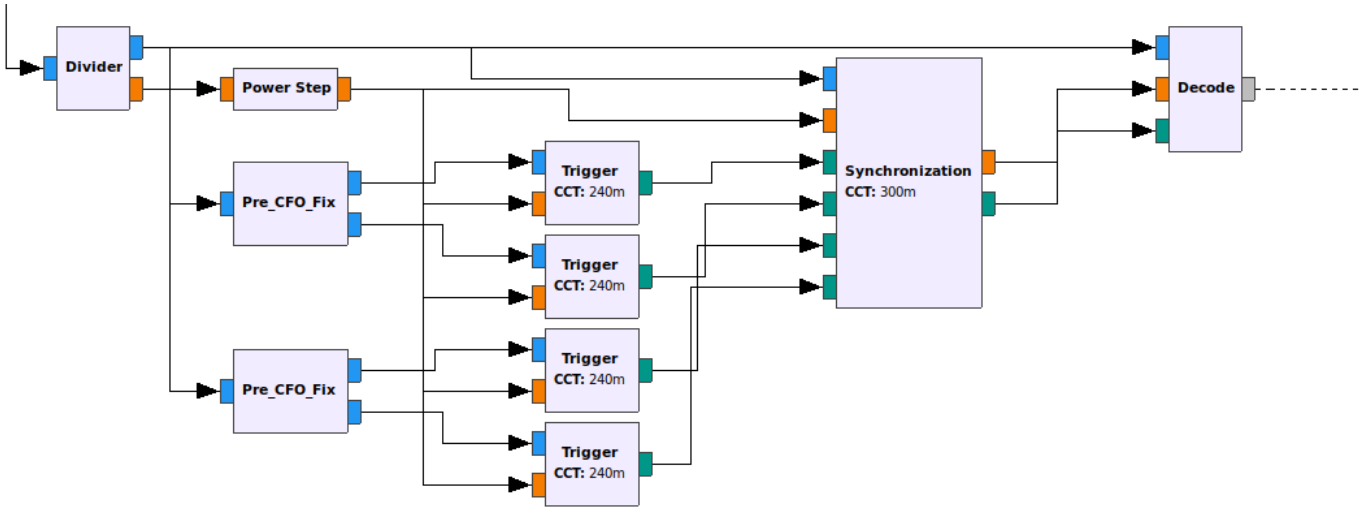

Fig. 8. The flowgraph of the proposed coherent OQPSK receiver.

- The `Decimating FIR Filter` module implements a matched filtering (MF) operation on the raw frequency estimates, with two filter taps $(1/2, 1/2)$.
- The `Low Pass Filter` (LPF) module aims to reduce the noise as much as possible before taking the phase difference of the incoming samples. The sampling rate for the GNU Radio implementation is 4 MHz, with the USRP baseband bandwidth set at 2 MHz. As the null-to-null frequency band of the incoming signal is $(-1.5, 1.5)$ MHz [9], we set the cutoff frequency at 1.5 MHz with a transition band of 0.1 MHz.

As we shall see in Section V-A, the MF module will bring about 4 to 5 dB performance gain while the LPF module will bring additional 1 dB performance improvement.

B. The Proposed Coherent receiver

The coherent receiver described in Section III has been implemented in the GNU Radio framework and its flowgraph is shown in Fig. 8. The code for the blocks can be found in the GitHub repository at [32]. There are six custom-made modules.

- The `Divider` module outputs the incoming complex data samples $x[n]$ and the squared magnitudes $|x[n]|^2$.
- The `Power Step` module computes the power sum $\sum_{i=1}^{N} |x[n-i]|^2$, which is needed by all correlation operations in other modules.
- The `Pre_CFO_Fix` module outputs a complex sequence after a pre-fixed CFO compensation. To speed up the processing, the pre-CFO value is chosen so that it divides the sampling rate f_s, i.e., $\epsilon_i = f_s/Q$ for an integer Q. This way, $e^{j2\pi\epsilon_i n t_s} = e^{j2\pi n/Q}$ repeats itself after Q samples and a finite-size look-up table can be used to

store the values. Also, the pre-CFO values ϵ_i and $-\epsilon_i$ are paired so that $e^{-j2\pi\epsilon_i t_s}$ are directly obtained from $e^{j2\pi\epsilon_i t_s}$ by conjugation.

- The `Trigger` module implements the triggering operation described in Section III-A and outputs $T_i[n]$ in (16). For the correlation operations, we downsample the correlation template and the received samples by two. The downsampled templates $\mathbf{d}_0, \mathbf{d}_7, \mathbf{d}_a$ have values of only $\pm 1, \pm i$, so no actual multiplication needs to be carried out when evaluating the inner products involved.
- The `Synchronization` module implements the synchronization and packet detection operation described in Section III-B, and outputs the timing and CFO estimates $(\hat{n}_{\mathrm{A7}}, \hat{\epsilon})$. The received samples are downsampled by two for the correlation operation in (28) and downsampled by 4 for the CFO estimation in (20).
- The `Decode` module implements the data detection module in Section III-C and outputs the decoded data packet. We have chosen the byte-by-byte update for the data demodulation. With simulations, we confirmed that the performance of the symbol-by-symbol update is nearly the same as the byte-by-byte update.

The parameters that can be customized by the users include: (a) the number of correlation branches; (b) the CFO values on the branches; (c) the detection threshold Γ_1 in the trigger module; and (d) the detection threshold Γ_2 in the synchronization module. Four correlation branches are used in Fig. 8 and hence two `Pre_CFO_Fix` and four `Trigger` modules are shown.

C. Computational Requirements

The computational requirements of different modules of the proposed receiver are analyzed below.

- The `Power Step` module requires two real multiplications and N real additions per incoming data sample. The `Pre_CFO_fix` module requires two complex multiplications per sample. The `Trigger` module has no complex multiplications in the correlators as the downsampled templates have only ± 1 values, but has $N/2$ complex additions and two real divisions. With $P = 4$ branches, the first stage of the receiver involves 4 complex multiplications, 2 real multiplications, $2P = 8$ real divisions, and $PN/2 = 128$ complex additions per incoming data sample.
- The `Synchronization` module is run once per detected packet. CFO compensation in (26) over 5 bytes in the preamble requires $5N = 320$ complex multiplications. Fine timing in (29) is typically accomplished within four samples. The operations in (29) and (30) have no more than $4 + 5 = 9$ byte-level correlations, which amounts to $2 \times 9N/2 = 576$ complex additions but no complex multiplications. The synchronization complexity is small relative to other modules as it occurs only once per data packet.
- The `Decode` module is run on the detected packet. Let us only track the number of complex multiplications here. The matrix inversion in (41) has complexity on the order of $\mathcal{O}(K^3)$, and is counted to have 255 complex multiplications in our current implementation. For each update in the byte-by-byte receiver, the numbers of complex multiplications corresponding to (35), (37), (40), (41) are $2N$, $2KN$, $2 \cdot 16 \cdot N$, $2NK^2 + 255 + 2NK + K^2$, respectively. With $N = 64$ and $K = 4$, there are a total of 5519 complex multiplications per byte update, which correspond to 43 complex multiplications per incoming data sample.

The MSK-based receiver has much lower complexity. The `Quadrature Demod` module has one complex multiplication per sample. The `Clock Recovery MM` runs on real samples and has 8 real multiplications per sample using the default 8-tap Minimum Mean Squared Error (MMSE) interpolator. The `Clock Recovery MM` outputs chip sequences at half the sampling rate. In the `Packet Sink` module, frame synchronization with 0xA7 uses 64 bit-level additions per chip. Once 0xA7 is found, the despreading operation runs 16 correlators with each correlator having 32 chips. To decode one byte of data, a total of $2 \cdot 16 \cdot 32 = 1024$ bit-level additions are needed, corresponding to $1024/128 = 8$ bit-level additions per incoming data sample.

To obtain an intuitive understanding, we put three receivers in one flowgraph and feed them with consecutive 30-byte data packets with 1 ms inter-packet separation. We access the `Performance Counters`, in particular the average clock cycles in call to the modules, through the use of `ControlPort` in the GNU Radio [33]. Based on the measurements, we infer that the enhanced MSK-based receiver requires approximately 2.7 times clock cycles relative to the original MSK-based receiver, which reduces to 1.4 times if the LPF is dropped (alternatively, the generic LPF module could be replaced by a custom-made module for speedup). On the other hand, the proposed receiver requires approximately 31 times clock cycles than the original MSK-based receiver. Note that the proposed receiver has not yet been optimized and a careful optimization could lead to multi-fold speedup. For example, the implementation of an IEEE 802.11 transceiver in [34] has been accelerated in [35] by a factor between $2\times$ and $10\times$ depending on the modulation and code scheme used.

V. Simulated Performance

We set the following parameters for the proposed receiver uploaded in [32]. There are a total of $P = 4$ branches with prefixed CFO values $\pm\frac{f_s}{250} = \pm 16$ kHz, $\pm\frac{f_s}{83} = \pm 48.2$ kHz. The detection thresholds are set as $\Gamma_2 = 0.3$ and $\Gamma_1 = 0.8\Gamma_2 = 0.24$. In Section V-A, we use the packet delivery ratio (PDR) as the performance metric to compare the coherent receiver and the MSK-based receivers. Other performance metrics are used to evaluate the coherent receiver in Section V-B.

A. PDR Performance of Different Receivers

Figs. 9 and 10 show the receiver performance for packets of 30 and 120 bytes in the presence of additive white Gaussian noise (AWGN), where the CFO value is uniformly drawn from the interval $[-64, 64]$ kHz. For each SNR point, several thousand packets are used for performance evaluation. The

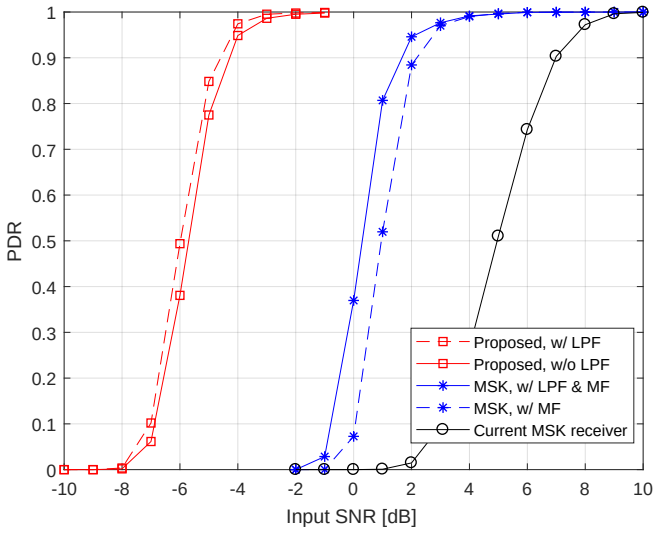

Fig. 9. PDR performance in AWGN, packet length: 30 bytes.

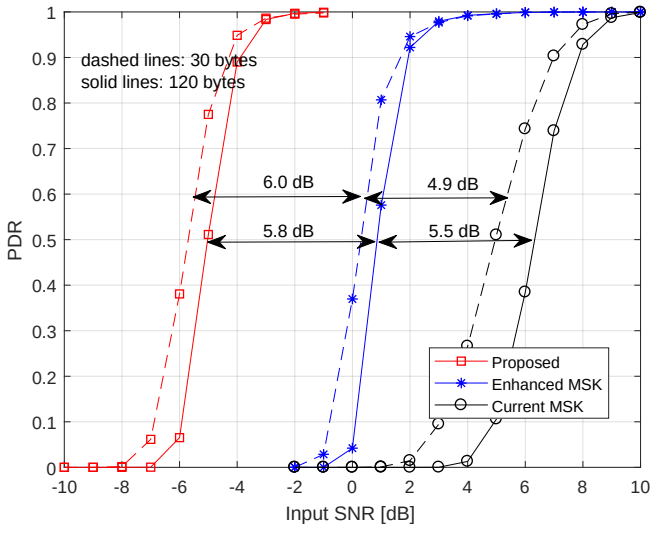

Fig. 11. PDR performance in AWGN.

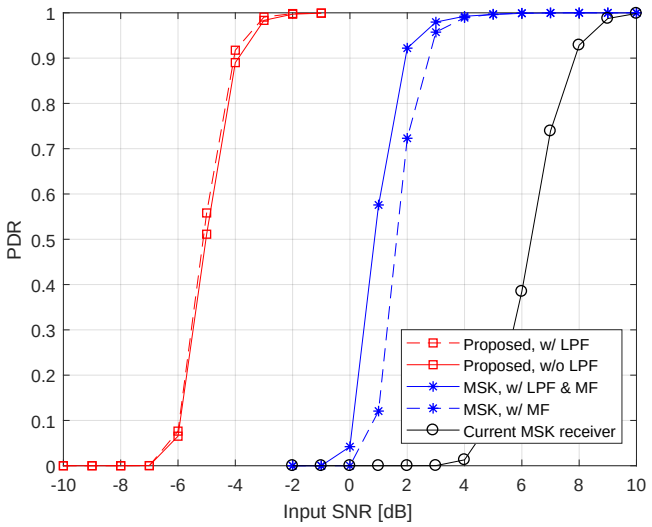

Fig. 10. PDR performance in AWGN, packet length: 120 bytes.

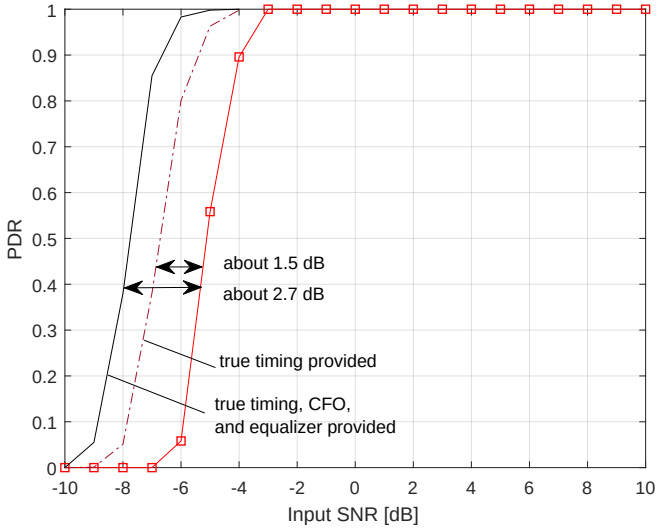

Fig. 12. Comparison with two genie-aided receivers, packet length: 120 bytes.

PDR curve of the original MSK-based receiver is consistent with that in Fig. 9 of [23], where an input SNR of 5 dB yields a PDR of 0.5 for packets with 20 bytes. We have made the following observations:

- For the modified MSK-based receiver, the MF module introduces 4 to 5 dB power gain and the LPF module introduces another 1 dB Gain.
- The coherent receiver with or without the LPF module has similar performance. Hence, the LPF module is bypassed in the GNU Radio package for the coherent receiver.

For convenience, we collect the PDR curves of the proposed receiver without LPF, the enhanced MSK receiver with both LPF and MF, and the original MSK receiver in Fig. 11, for packets of 30 and 120 bytes. We observe that the enhanced MSK receiver has about 5 dB gain over the original MSK receiver, while its performance lags behind the proposed coherent receiver by about 6 dB.

The performance gain of the proposed coherent receiver over the MSK receiver in [21] is larger than we expected at the beginning of this work. It is well-known that antipodal signalling (e.g., binary phase shift keying) outperforms binary orthogonal signalling (e.g., binary frequency shift keying) by 3 dB and coherent decoding of binary orthogonal signals outperforms noncoherent decoding by about 1 dB. Hence, we started the work expecting a 4 dB improvement resulting from coherent decoding of OQPSK over non-coherent decoding of MSK. However, there is a large spreading gain in the IEEE 802.15.4 OQPSK physical layer. The proposed coherent receiver first establishes carrier and time synchronization, and the demodulator works well even when the input SNR is well below 0 dB due to the large spreading gain. On the other hand, the key step in the MSK-based receiver is to take the phase difference of consecutive received samples $x[n]$, which enables subsequent low-complexity symbol syn-

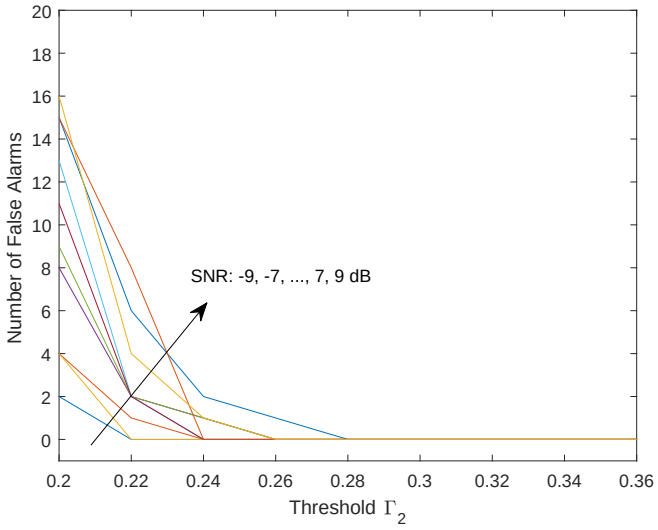

Fig. 13. Number of false alarms for data segments of 1 ms long.

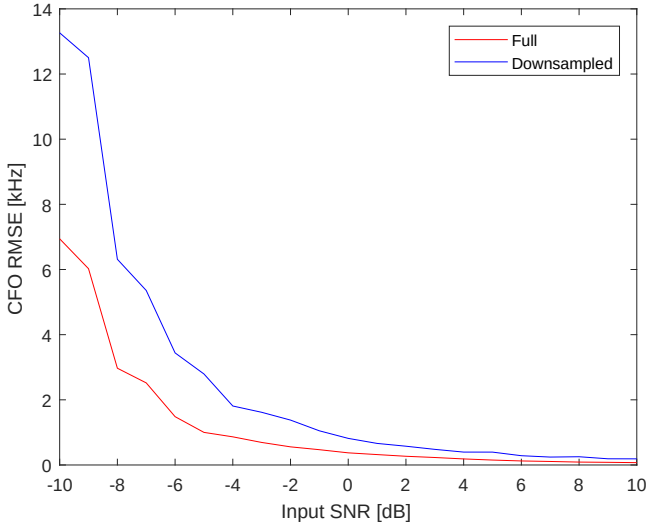

Fig. 14. Accuracy of the CFO estimation.

chronization and data detection. Let $x[n] = s[n] + w[n]$, where $s[n]$ is the signal and $w[n]$ is the noise. The operation $\angle\big((s[n-1]+w[n-1])^*(s[n]+w[n])\big)$ works well only when the signal is reasonably stronger than the noise, otherwise the phase estimate will correspond to the noise rather than the signal. This step operates on a per-sample basis, so it does not benefit from the redundancy in the modulation, and hence is one key performance bottleneck for the MSK based receiver.

In Fig. 12, we compare the proposed receiver with two genie-aided receivers for packets of length 120 bytes. In both genie-aided receivers, all packets are correctly detected. The first receiver assumes perfect timing $\hat{n}_{\mathrm{A7}} = n_{\mathrm{A7}}$ but with practical CFO estimation and data-directed equalizer update, while the second receiver assumes perfect timing $\hat{n}_{\mathrm{A7}} = n_{\mathrm{A7}}$, perfect CFO estimation $\hat{\epsilon} = \epsilon$ and the optimal equalizer $\mathbf{f}[l] = [0, 1, 0, 0]$ for an AWGN channel. The second receiver corresponds to the receiver analyzed in [9], which assumes perfect carrier and time synchronization and tests only the demodulation performance in an AWGN channel. Since every four bits are mapped into 64 samples, E_{b}/N_0 is $10\log_{10}(64/4) = 12$ dB higher than the input SNR. Fig. 4 of [9] shows that a symbol error rate (SER) of 10^{-4} is achieved at $E_{\mathrm{b}}/N_0 = 7$ dB and hence SNR $= -5$ dB, and the corresponding PDR is $(1 - \mathrm{SER})^{120\cdot 2} = 0.98$ for a 120-byte packet. Our result in Fig. 12 is hence consistent with the union bound and the simulated performance in Fig. 6 of [9].

From Fig. 12, we observe that the proposed receiver is about 2.7 dB away from the ideal receiver with perfect carrier and time synchronization in an AWGN channel [9]. Out of the 2.7 dB gap, 1.5 dB is attributed to the CFO estimation and data detection modules and 1.2 dB is attributed to the packet detection and time synchronization modules.

B. Detection Threshold, CFO and SNR Estimation

Here we report other performance metrics for the proposed coherent receiver: packet detection performance, CFO estimation accuracy and SNR estimation.

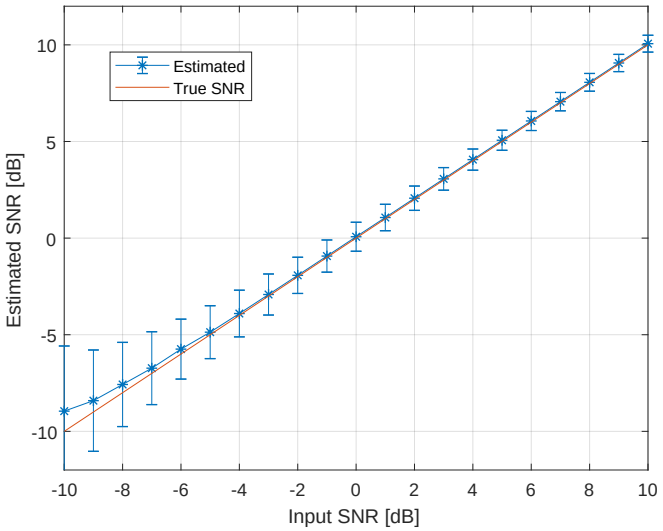

Fig. 15. The estimated SNR as a link quality indicator.

Packet detection has two probabilities to deal with: Probability of detection and probability of false alarms. Lower detection thresholds can improve the probability of detection but will increase the probability of false alarms. We choose the constant as

$$\Gamma_1 = 0.8\,\Gamma_2 \tag{43}$$

and vary Γ_2 between 0.2 and 0.4. It turns out that the false alarms due to additive white Gaussian noise can be neglected in view of the randomly generated data packets. Fig. 13 depicts the joint effect of mixing randomly generated packets (without the preamble portion) with additive noise on the number of false alarms processing each data segment of 1 ms (corresponding to a packet of 30 bytes). When $\Gamma_2 = 0.3$ and $\Gamma_1 = 0.8\Gamma_2 = 0.24$, false alarms are negligible. We hence recommend these threshold values for the coherent receiver.

For the correctly detected packets, the root mean-squared error (RMSE) of the CFO estimates is shown in Fig. 14,

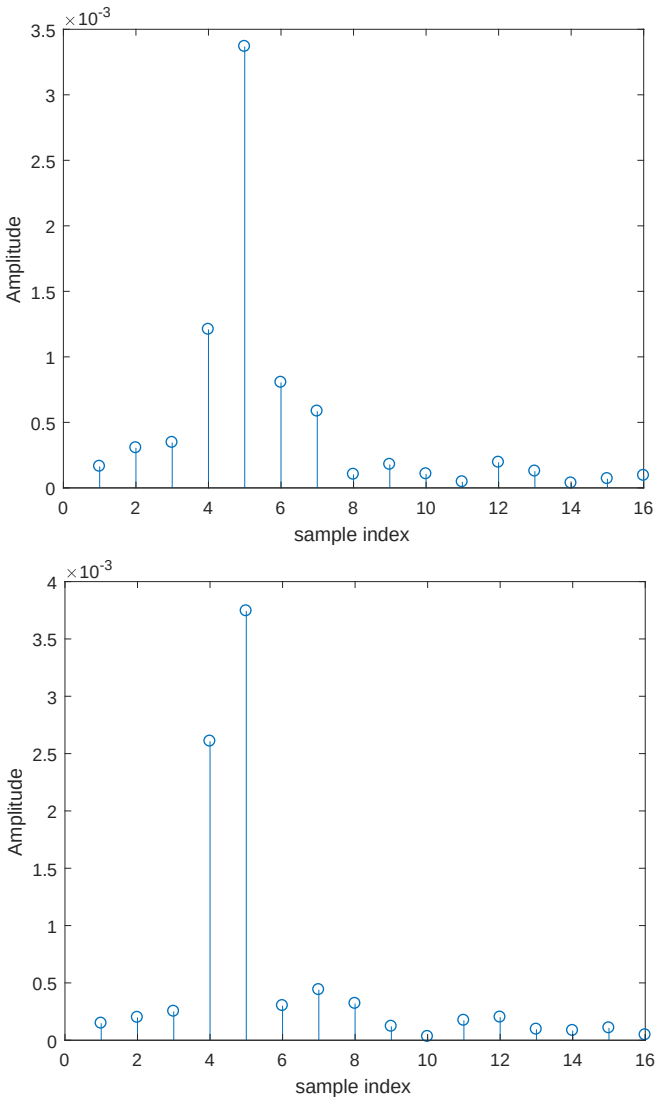

Fig. 16. Example channel impulse responses: Sample interval $t_s = 0.25\ \mu s$.

where the original data sequence could be also downsampled by four times to speed up the processing. As the input SNR stays above -6 dB, the RMSE is below 4 kHz. The average values of the SNR estimates are depicted in Fig. 15 with the standard deviation illustrated via error bars around the mean. The average SNR agrees with the true SNR very well. The standard deviation of the SNR estimates is about 1.5 dB at input SNR of -5 dB and is about 0.75 dB at input SNR of 0 dB. The SNR estimate can serve as one reliable link quality indicator, especially when averaged over several packets.

C. Emulated Performance based on Recorded Data Sets

We used USRP to record several data sets from the transmissions of a TI CC2650 device. The estimated CFO between the transmitter and the receiver is about 10.5 kHz. Channel estimates based on the least-squares fitting of the received samples corresponding to the 0xA7 byte are plotted in Fig. 16, and are similar to those reported in [36]. These channel plots motivated the choice of the four-tap equalizer with $K_1 = 1$ and $K_2 = 2$.

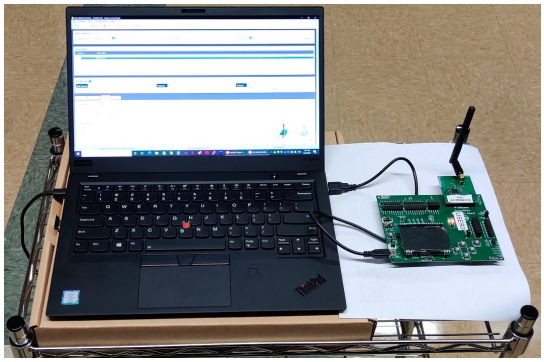

Fig. 17. The TI CC2650 transmitter on a moving cart.

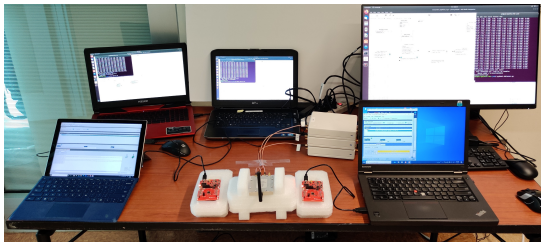

Fig. 18. Five receivers placed side by side: three GNU Radio receivers and two TI CC2652 receivers.

VI. Field Test

We performed a field test in an indoor environment. Fig. 17 shows a TI CC2650 transmitter [37], placed on a cart. Fig. 18 shows the receiver setup with five receivers placed together. One monopole antenna is connected to a divider which provides identical RF signals to three USRP units through wired connections. With samples from USRP devices, three GNU Radio receivers were running: a) the proposed coherent receiver, b) the enhanced MSK-based receiver and c) the original MSK-based receiver. Two CC2652 receivers from Texas Instruments [38] were placed near the monopole antenna, one on the left and one on the right. The CC2652 receivers have their own printed-circuit board (PCB) inverted-F antennas, whose beam patterns can be found in [39]. The normalized receiver gains on the USRP units were set to 0.9. To avoid interference from the WiFi, we used IEEE 802.15.4 channel 20 centered at 2.450 GHz.

The third floor of the Information Technology and Engineering (ITE) building at the UConn campus, whose floor map is shown in Fig. 19, was used for tests. At points P1, P2, P3, the transmitter cycled the power levels from the set of ten values $\{-21, -18, -15, \ldots, 0, 3, 5\}$ dBm, where 5 dBm is the maximum power allowed by the device. Point P1 is 28 meters away and has a direct line-of-sight (LOS) path to the receiver. P2 and P3 are 40 and 49 meters away from the receiver, respectively. Although there are open hallways between P2/P3 and the receiver, the transmission paths might not be claimed as LOS paths. All receivers decoded nearly all the packets at P1 for all power levels. Figs. 20 and 21 show the PDR performance at P2 and P3, respectively. For each PDR evaluation, 500 packets were sent with packet length 30 bytes. From Figs. 20 and 21, we observe that the proposed

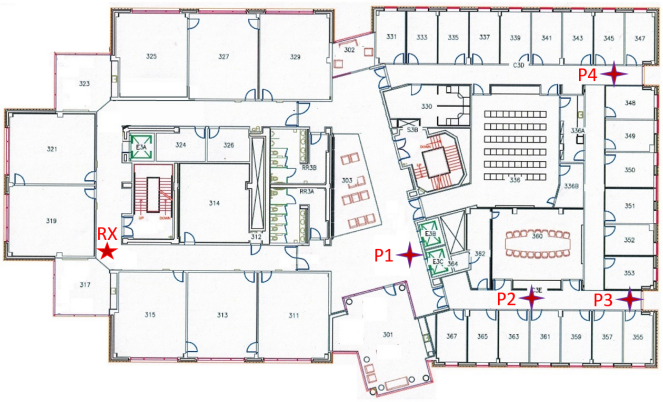

Fig. 19. The floor plan with marked receiver and transmitter locations.

receiver has approximately a 6 dB gain over the enhanced MSK-based and a 10 dB gain over the original MSK-based receiver. These performance gains are mostly consistent with the simulation results. The TI receivers are worse than the MSK-based receiver and show large performance variations between P2 and P3. However, the TI receivers and the USRP receivers do not share the same antennas and the antenna patterns have an influence that cannot be quantified here. In addition, we tried to transmit packets at point P4, which is on the other side of the hallway and has transmission paths blocked from the receiver. At a transmit power of 5 dBm, the proposed coherent receiver received 80% of the packets while all other receivers received zero packets.

We further examine the SNR estimates provided by the coherent receiver (the "debug" option in the `Decode` module in Fig. 8 needs to be turned on). The average SNRs at P1-P4 are reported in Fig. 22 as a function of transmit power, and the standard deviation is about 1.6 dB. Moving the transmitter from P1 to P2, nearly 10 dB SNR is lost, while P2 and P3 have almost the same SNRs. Corresponding to the transmit power of 5 dBm, the average SNR is only -2.7 dB at P4, while about 17 dB at P3. The SNR estimates shed light on the PDR performance differences in Fig. 20 and 21. We underscore that indoor propagation channels are complex and the receiver performance is not merely a function of the transmission distance.

VII. Conclusion

In this paper, we presented a coherent receiver for the IEEE 802.15.4 OQPSK physical layer and implemented it in the GNU Radio framework. The receiver achieves a significant improvement around 11 dB in performance over an existing GNU radio receiver, and maintains about 6 dB gain when the latter is suitably improved. This proposed receiver is suitable for SDR-based high-performance gateways in IoT applications. Our future work will include code optimization for processing speeds and application of the proposed receiver in IoT testbeds.

Authors' Contributions

E. Faulkner and Z. Yun have contributed equally to this work. E. Faulkner focused on algorithm development and performance analysis, while Z. Yun implemented the GNU radio receiver and led the experimental validation.

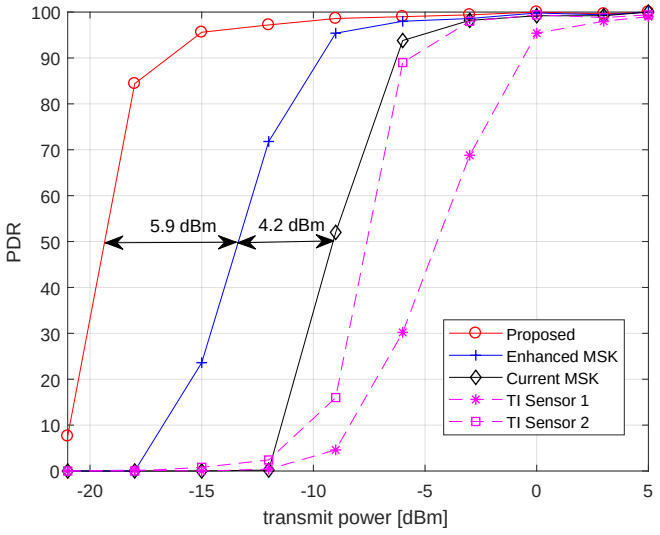

Fig. 20. PDR vs transmission power at P2.

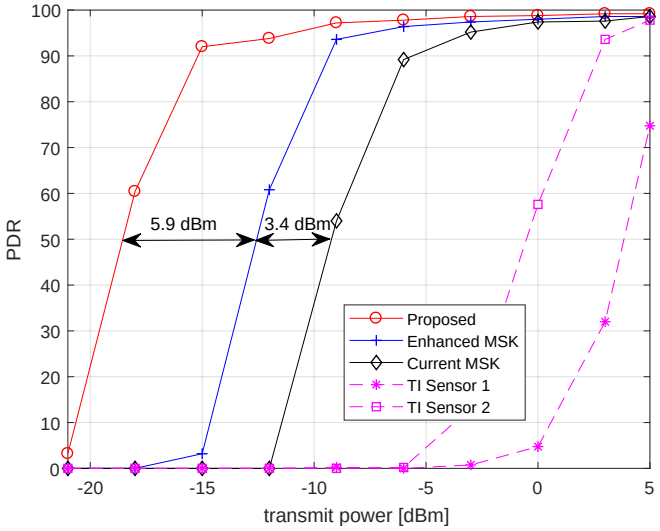

Fig. 21. PDR vs transmission power at P3.

Acknowledgement

The work by Z. Yun and S. Han is partially supported by the National Science Foundation under NSF Awards CNS-1925706 and IIP-1919229. We thank Ms. Natong Lin for her help in assisting the experimental tests, and Kaiyuan Liu for his help in investigating the receiver complexity. We are grateful to Dr. Bloessl whose open-source GNU-radio receiver in [21] has motivated this work.

References

[1] *IEEE Standard for Low-Rate Wireless Networks*, IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011) Std., 2016.

[2] A. G. Ramonet and T. Noguchi, "IEEE 802.15.4 historical evolution and trends," in *Prof. of 21st International Conference on Advanced Communication Technology (ICACT)*, 2019, pp. 351–359.

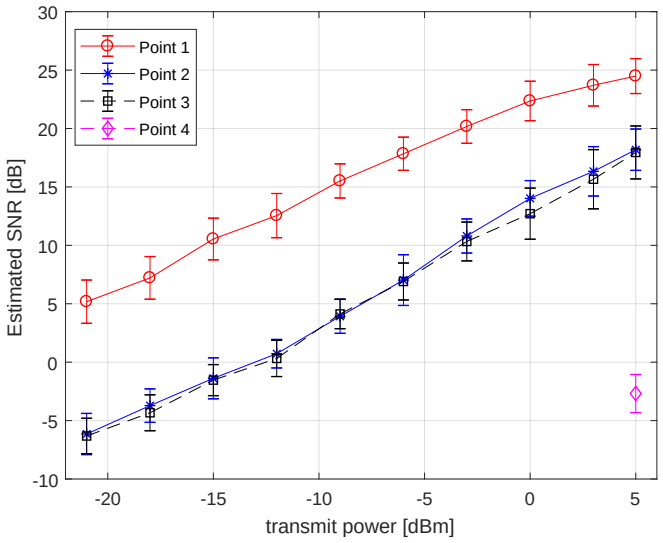

Fig. 22. The estimated SNRs at P1, P2, P3, P4.

[3] L. Davoli, L. Belli, A. Cilfone, and G. Ferrari, "From Micro to Macro IoT: Challenges and solutions in the integration of IEEE 802.15.4/802.11 and Sub-GHz technologies," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 784–793, 2018.

[4] N. Choudhury, R. Matam, M. Mukherjee, and L. Shu, "Beacon synchronization and duty-cycling in IEEE 802.15.4 cluster-tree networks: A review," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1765–1788, 2018.

[5] W. Li, X. Hu, and T. Jiang, "Path loss models for IEEE 802.15.4 vehicle-to-infrastructure communications in rural areas," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3865–3875, 2018.

[6] M. Wu, X. Hu, R. Zhang, and L. Yang, "Collision recognition in multihop IEEE 802.15.4-compliant wireless sensor networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8542–8552, 2019.

[7] G. Chen, X. Cao, L. Liu, C. Sun, and Y. Cheng, "Joint scheduling and channel allocation for end-to-end delay minimization in industrial WirelessHART networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2829–2842, 2019.

[8] J. G. Proakis, *Digital Communications*. McGraw-Hill,4th edition, 2001.

[9] P. Gupta and S. G. Wilson, "IEEE 802.15.4 PHY analysis: Power spectrum and error performance," in *Prof. of Annual IEEE India Conference*, vol. 1, 2008, pp. 171–176.

[10] D. Park, C. S. Park, and K. Lee, "Simple design of detector in the presence of frequency offset for IEEE 802.15.4 LR-WPANs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 4, pp. 330–334, 2009.

[11] S. Dai, H. Qian, K. Kang, and W. Xiang, "A robust demodulator for OQPSK-DSSS system," *Circuits, Systems, and Signal Processing*, vol. 34, pp. 231–247, 01 2014.

[12] U. Pešović, S. Durašević, and P. Planinšiš, "Carrier synchronization algorithm for software defined radio," in *Proc. of 25th Telecommunication Forum (TELFOR)*, 2017, pp. 1–4.

[13] K. Gorantla and V. V. Mani, "Synchronization in IEEE 802.15.4 Zigbee transceiver using Matlab Simulink," in *Proc. International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 144–148.

[14] S. Yin, J. Cui, A. Luo, L. Liu, and S. Wei, "A highly efficient baseband transceiver for IEEE 802.15.4 LR-WPAN systems," in *Prof. of 9th IEEE International Conference on ASIC*, 2011, pp. 224–227.

[15] U. Pešović, D. Gliech, P. Planinšič, Z. Stamenković, and S. Randić, "Implementation of coherent IEEE 802.15.4 receiver on software defined radio platform," in *Prof. of 23rd Telecommunications Forum Telfor (TELFOR)*, 2015, pp. 224–227.

[16] R. M. Koteng, "Evaluation of SDR-implementation of IEEE 802.15.4 physical layer," Master's thesis, Norwegian University of Science and Technology, 2006.

[17] A. Z. Mohammed, A. K. Nain, J. Bandaru, A. Kumar, D. S. Reddy, and R. Pachamuthu, "A residual phase noise compensation method for IEEE 802.15.4 compliant dual-mode receiver for diverse low power IoT applications," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3437–3447, 2019.

[18] N. Dehaese, S. Bourdel, H. Barthelemy, and G. Bas, "Simple demodulator for 802.15.4 low-cost receivers," in *Prof. of IEEE Radio and Wireless Symposium*, 2006, pp. 315–318.

[19] T. Schmid, "GNU Radio 802.15.4 En- and Decoding," University of California Los Angeles, Tech. Rep., 2006.

[20] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer, "A GNU Radio-based IEEE 802.15.4 Testbed," in *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, Cottbus, Germany, September 2013, pp. 37–40.

[21] B. Bloessl, "Wirless Measurement and Experimentation (WIME)," https://www.wime-project.net/.

[22] R. Zitouni, L. George, and Y. Abouda, "A Dynamic Spectrum Access on SDR for IEEE 802.15.4e networks," in *Proceedings of Wireless Innovation Forum*, March 2015.

[23] B. Bloessl and F. Dressler, "msync: Physical layer frame synchronization without preamble symbols," *IEEE Transactions on Mobile Computing*, vol. 17, pp. 2321–2333, 2018.

[24] H. K. Benitez, C. H. Cabuso, M. T. De Leon, J. R. Hizon, and M. Rosales, "Implementation of a physical layer wireless sensor network testbed using software defined radios," in *Prof. of International Symposium on Multimedia and Communication Technology (ISMAC)*, 2019, pp. 1–6.

[25] J. Gu, C. Chen, S. Zhu, and J. He, "Efficient Error Packet Recovery without Redundant Bytes for IEEE 802.15.4 Protocol," in *3rd International Symposium on Autonomous Systems (ISAS)*. Shanghai, China: IEEE, May 2019, pp. 418–423.

[26] S. Gvozdenovic, J. K. Becker, and D. Starobinski, "SDR-based PHY Characterization of Zigbee Devices," in *63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. Springfield, MA: IEEE, August 2020, pp. 129–132.

[27] S. Wang, Y. Li, C. Ming, and Z. Zhang, "Building Gateway Interconnected Heterogeneous ZigBee and WiFi Network Based on Software Defined Radio," in *International Conference on Communications and Networking in China (ChinaCom)*. Shanghai, China: Springer, November 2019, pp. 445–456.

[28] C. Gavril, C.-Z. Kertesz, M. Alexandru, and V. Popescu, "SDR-based Gateway for IoT and M2M applications," in *Prof. of Baltic URSI Symposium (URSI)*. Poznan, Poland: IEEE, May 2018, pp. 71–74.

[29] H. Hellstrom, M. Luvisotto, R. Jansson, and Z. Pang, "Software-defined wireless communication for industrial control: A realistic approach," *IEEE Industrial Electronics Magazine*, vol. 13, no. 4, pp. 31–37, 2019.

[30] R. C. Johnson Jr, W. A. Sethares, and A. G. Klein, *Software Receiver Design: Build Your Own Digital Communication System in Five Easy Steps*. Cambridge University Press, 1st edition, 2011.

[31] K. Mueller and M. Müller, "Timing recovery in digital synchronous data receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 516–531, 1976.

[32] https://github.com/cloud9477/gr-advoqpsk.

[33] T. Rondeau, T. O'Shea, and N. Goergen, "Inspecting GNU Radio Applications with ControlPort and Performance Counters," in *Proc. of 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum*, Hong Kong, China, August 2013, pp. 65–70.

[34] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM Receiver for GNU Radio," in *Proc. of 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum*, Hong Kong, China, August 2013, pp. 9–16.

[35] G. Arcos, R. Ferreri, M. Richart, P. Ezzatti, and E. Grampín, "Accelerating an IEEE 802.11 a/g/p Transceiver in GNU Radio," in *Proc. of 9th Latin America Networking Conference (LANC'16)*, Valparaíso, Chile, October 2016, pp. 13–19.

[36] S. Ayvaşik, M. Gürsu, and W. Kellerer, "Veni Vidi Dixi: reliable wireless communication with depth images," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, Orlando, FL, USA, December 912, 2019.

[37] Texas Instruments, "SimpleLink 32-bit Arm Cortex-M3 multiprotocol 2.4 GHz wireless MCU," https://www.ti.com/product/CC2650.

[38] ——, "SimpleLink multi-standard CC26x2R wireless MCU LaunchPad development kit," https://www.ti.com/tool/LAUNCHXL-CC26X2R1.

[39] ——, "Application Report: 2.4-GHz Inverted F Antenna," https://www.ti.com/lit/an/swru120d/swru120d.pdf.