Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/5524--15-9575

A Constant Envelope OFDM Implementation on GNU Radio

Andrew Robertson
Amos Ajo
Sastry Kompella
Joe Molnar

Networks and Communications Branch
Information Technology Division

Frank Fu

KEYW Corporation
Hanover, Maryland

February 2, 2015

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
02-02-2015	Memorandum Report	

4. TITLE AND SUBTITLE

A Constant Envelope OFDM Implementation on GNU Radio

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Andrew Robertson, Amos Ajo, Sastry Kompella, Joe Molnar, and Frank Fu*

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory, Code 5524
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5524--15-9575

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Naval Research Laboratory, Code 5524
4555 Overlook Avenue, SW
Washington, DC 20375-5320

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

*KEYW Corporation, 7740 Milestone Pkwy, Suite 400, Hanover, MD 21076

14. ABSTRACT

Distributed sensor networks often call for bursty, high data rate communications over long distances and fluctuating channels. A waveform for such a scenario would ideally be wideband to mitigate frequency selective fading, have data multiplexed into separate streams for high data rate, have long symbol times to mitigate multipath, and have a constant amplitude over time for non-linear amplification. These requirements are uniquely served by constant envelope OFDM. We describe the use-cases, theory, and successful implementation of such a waveform via the GNU radio framework on Ettus software-defined radios. We measure the bit error performance of the link as a function of channel quality and compare this performance to simulations of the link.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Andrew Robertson
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	Unclassified Unlimited	23	**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified Unlimited	Unclassified Unlimited	Unclassified Unlimited			(202) 767-3262

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

CONTENTS

A CONSTANT ENVELOPE OFDM IMPLEMENTATION ON GNU RADIO

1. BACKGROUND

We wish to consider a waveform that supports communications between spatially separated sensors. Because sensors ideally spend most of their time sensing, we would like a waveform that communicates in short bursts (tens of milliseconds) on platforms that are not capable of simultaneous transmit and receive. The short times allowed for transmission require that overhead carrier acquisition be minimized. Thus, incoherent modulation is preferable to one that relies on knowing the precise phase and frequency of the transmitted carrier. We also wish to consider a fluctuating channel such that channel estimation will have to be done at the beginning of every data payload by means of a short pilot pulse. This also aids synchronization at sample time resolution so that the receiver can decide the precise time limits of the symbols so that it can perform its correlations with known symbol signals.

The design of high data-rate RF data communication links over long-distance fluctuating channels presents a complex trade-space. The high data-rate requirement necessitates either short symbol times or parallel data streams. The symbol times are determined by two competing interests: the required data rate and the resilience to multipath fading. We would like to use the longest symbol time that will reliably guarantee the required data-rate. Longer symbol times yield an integration gain that protects against multipath and noise, but they require parallelization of the data stream. Short symbol times cause higher error rates because the receiver has less time to average away noise. This problem is compounded by multipath delayed signal components resulting in heavy inter-symbol interference (ISI). On the other hand, data stream parallelization techniques such as OFDM come with their own set of problems. The greatest of these is the high peak-to-average power ratio (PAPR) of parallelized waveforms. That is, any waveform that can be represented as a sum of separately modulated sub-carriers will have large fluctuations in its transmitted amplitude over time. Because power amplifiers have only a finite range in input amplitude over which their gains are linear, the waveform must either be input at artificially low power (by an amount called the *back-off*) or face non-linear distortion of the waveform resulting in symbol errors. Many operational scenarios would benefit from a waveform that had long symbol times for resilience against noise, parallelization of streams over separate subcarriers for high data rate, and constant amplitude for high amplifiability.

The great distance between sensors necessitates high amplification, but intermodulation distortions arise when the signal input to an amplifier is a sum of modulated subcarriers [1]. Such distortions can be completely nullified by using a constant-amplitude modulation (FSK, PSK, GMSK, etc.), but this is a heavy restriction because a sum of constant-amplitude waveforms is no longer constant-amplitude. Thus, such modulations often cannot be parallelized to send more data over a large bandwidth. The combination of high data rate and long symbol time requirements means that the waveform must be multiplexed across a large bandwidth like OFDM. However nonlinear amplification precludes the use of OFDM and any other modulation that is a sum subcarriers because they are not constant amplitude.

Manuscript approved December 10, 2014.

These niche requirements necessitate a waveform that is incoherent, bursty, high-instantaneous data rate, constant amplitude, and has a long symbol time. Constant envelope OFDM (CE-OFDM) is an experimental wave-form that fits these requirements [2–5]. In this waveform, the phase of a single carrier is modulated with an OFDM signal. Like OFDM, the modulation parallelizes a serial stream of data into several streams that modulate separate subcarriers. Unlike OFDM, these modulated subcarriers are summed and written into the phase of a single RF carrier so that the resulting signal has constant amplitude. Due to the parallelization, CE-OFDM is wideband and high data rate. It is constant amplitude, continuous phase, and it can be incoherently received.

The goal of this project is to produce a CE-OFDM waveform on a GNU Radio platform and validate its properties. This means measuring the bit-error performance as a function of noise conditions, channel conditions (including different multipath amplitude-delay profiles), channel equalization strategies, packet frame parameters, and amplification hardware. This report details our successful implementation of the waveform on GNU Radio and where we are presently on our course toward waveform validation.

2. THEORY OF CE-OFDM OPERATION

2.1 CE-OFDM Transmission

Let each data payload consist of a sequence of N_{bits} bits. The data is written in non-zero returning (NZR) form as the sequence $\{d_l\}_{l=1}^{N_{\text{bits}}} = \{\pm 1, \pm 1, \ldots, \pm 1\}$. Our first step is to parallelize the data into separate streams that will modulate the OFDM subcarriers.

$$\{d_l\}_{l=1}^{N_{\text{bits}}} = \{\pm 1, \pm 1, \ldots, \pm 1\} \rightarrow \begin{pmatrix} \pm 1 & \pm 1 & \cdots \\ \pm 1 & \pm 1 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} = [I_{i,k}]_{N_{\text{subcarriers}} \times N_{\text{symbols}}} \tag{1}$$

The length N_{bits} serial stream of data has been converted to $N_{\text{subcarriers}}$ parallel streams of N_{symbols} symbols where (assuming the d_l to be binary) $N_{\text{bits}} = N_{\text{subcarriers}} \times N_{\text{symbols}}$. These bits modulate an orthogonal set of $N_{\text{subcarriers}}$ subcarriers. The subcarriers are functions $q_n(t)$ that are only nonzero over the time domain $0 \leq t < T_S$ where T_S is the symbol time. We choose them to have the property that $\int_0^{T_S} dt\, q_m(t)\, q_n(t) = \lambda \delta_{m,n}$ for some constant λ so that we can design a suboptimal receiver with close to optimal symbol error rate. To guarantee a continuous phase between symbols and their cyclic prefixes (for spectral containment), we would like to use a set of subcarriers for which $q_m(t)$ approaches 0 as $t \rightarrow T_S^-$ and $t \rightarrow 0^+$. The family of half-sines satisfies this requirement, but that they add a phase distortion proportional to $\int_0^{T_S} dt\ q_m(t)$ [2]. To maintain phase continuity without the phase distortion, we use only the even half-sines. The distortion integral vanishes for these functions. Letting p be some *even* integer, we have:

$$q_m(t) = \begin{cases} sin(\pi pmt/T_S) & 0 \leq t < T_S \\ 0 & \text{else} \end{cases} \tag{2}$$

In our work so far, we use $p = 2$. This integer is one of the knobs we can tune to increase the bandwidth of the waveform and provide more frequency diversity to mitigate the frequency selective fading of the channel. Greater values of p result in greater waveform bandwidth.

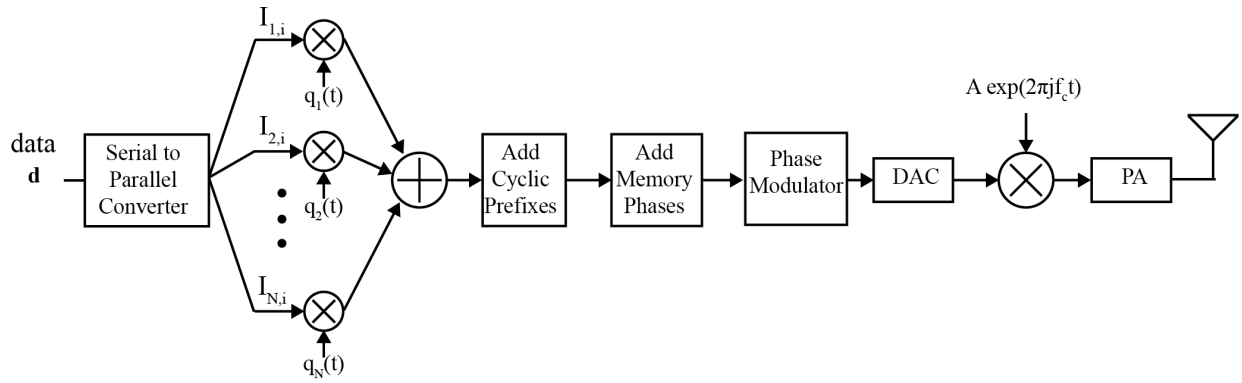

Fig. 1 — The CE-OFDM transmission chain for real samples. Real samples require a phase modulator separate from the DAC.

We now define a baseband OFDM waveform that will modulate the phase of an RF carrier. The OFDM signal for a block of N_{symbols} symbols can be written as $v(t)$. The subcarrier multiplication and summing operations in Fig. (1) have the form:

$$v(t) = 2\pi hC \sum_{i=1}^{N_{\text{subcarriers}}} \sum_{k=1}^{N_{\text{symbols}}} \left\{ I_{i,k}\, q_i[t-(k-1)\,T_S] + \theta_k \right\} \tag{3}$$

We will discuss the memory phases θ_k later. The constant h is called the *modulation index* which scales the amplitude of the OFDM signal and sets the phase variance of the CE-OFDM signal. The normalizing constant C ensures that the phase variance of the CE-OFDM signal is a function of h only. It is written in terms of the variance σ_d^2 of the information signal $\{d_l\}$. For equally likely independent symbols from an alphabet of size M, the symbol variance is $\sigma_d^2 = \frac{M^2-1}{3}$. Since $M = 2$ for binary $\{d_l\}$, we have

$$C = \sqrt{\frac{2}{N_{\text{subcarriers}}\,\sigma_d^2}} = \sqrt{\frac{2}{N_{\text{subcarriers}}}} \tag{4}$$

Equation (3) is an OFDM signal, so it is not contant-amplitude. While parallelization of the data stream allows for the communication of a lot of data in a short time over a large bandwidth, the $v(t)$ is still a sum of modulated subcarriers. Worse, the data signal modulates the amplitudes of those subcarriers. This information will be heavily corrupted by the non-linear distortion of the amplifier. To mitigate this, we must take the crucial step of writing this OFDM signal into the phase of a single carrier. Letting $j = \sqrt{-1}$ the resulting base-band CE-OFDM signal $s(t)$ is:

$$s(t) = e^{jv(t)} \tag{5}$$

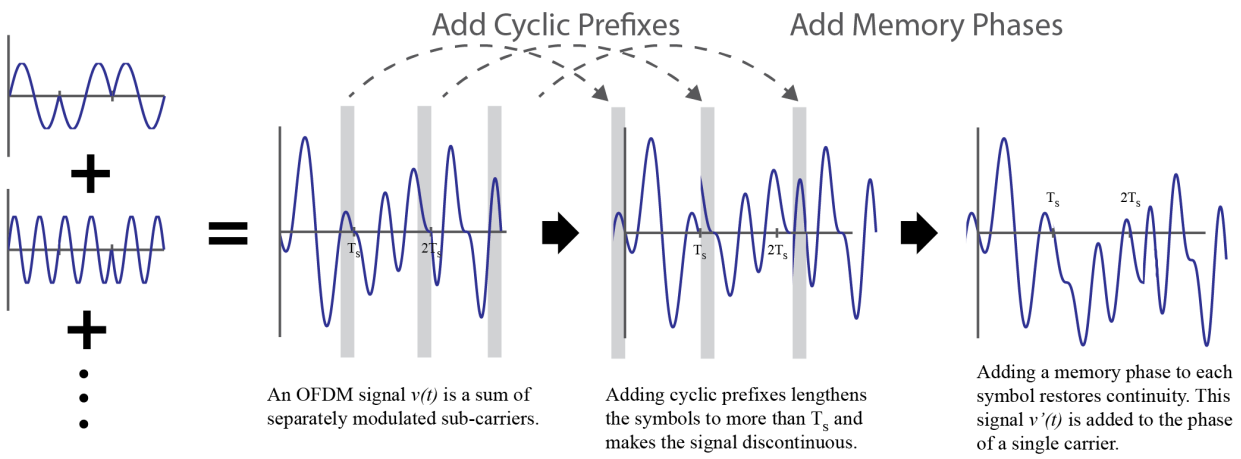

Fig. 2 — The initial phase signal described by $\nu(t)$ is continuous. The addition of cyclic prefixes destroys this, but memory phases are added so that $\nu'(t)$ is continuous.

The last step that is strictly necessary is to multiply $s(t)$ by the carrier and transmit. However we elect to add a cyclic prefix to each symbol so that the effect of the multipath channel can be reversed in the frequency domain. We then choose our memory phases θ_k to guarantee phase continuity across symbol thresholds. The memory phases have the added effect that information is encoded in the *time-derivative* of the phase rather than the absolute phase during the symbol time. This obviates the need for the receiver to be phase-locked to the transmitter. The modulation becomes incoherent, and we save time that would have been required for carrier acquisition. However the addition of cyclic prefixes must be done explicitly in the digital domain. Assuming a sampling rate f_s and letting $t_s = f_s^{-1}$, let us define the signal we have at this stage.

$$s_n = s(nt_s) \tag{6}$$

This sequence consists of N_{symbols} symbols in succession each of which is itself a sequence of $N = f_s T_S$ complex samples. We know that the effect of a multipath channel will be to convolve these samples with the channel's impulse response. If we add the final L samples of each symbol to the beginning of that symbol, then the effect of the channel on the symbols becomes a *circular* convolution. The cyclic prefixes contain only redundant information and will be discarded after the signal is sampled at the receiver. Their only function is to make the convolution with the channel impulse response circular instead of linear. This is an important function because circular convolutions are diagonalized by discrete Fourier transforms. Thus we can reverse the effect of the channel in the frequency domain [6]. To add a cyclic prefix of length L to symbol i, we take the L final samples of the symbol and append them in the same order to the beginning of the symbol. That is, we take the samples $\{s_{iN-L}, s_{iN-L+1}, \ldots, s_{iN-1}\}$ and append them to the beginning of symbol i creating a longer symbol with samples $\{s_{iN-L}, s_{iN-L+1}, \ldots, s_{iN-1}, s_{(i-1)N}, s_{(i-1)N+1}, \ldots, s_{iN-1}\}$. This new symbol has length $N' = L+N$ and duration $T_S' = (L+N)t_s$, but L samples contain redundant information. The price we pay for lower bit-error rates due to channel distortion is a data rate lowered by a factor of $N/(N+L)$. Consequently, we want L to be as small as possible such that the channel can be reliably equalized. The cyclic prefixes should be slightly longer than the multipath delay.

Table 1 — Digital Operations per Symbol in the Transmit Chain

Signal Operation	**Mathematical Operation**
Store data vector	Store $1 \times N_{\text{bits}}$ vector
Serial-to-Parallel Converter	Reshape $1 \times N_{\text{bits}}$ vector to $N_{\text{subcarriers}} \times N_{\text{symbols}}$ matrix
Storage of subcarriers	Store $N_{\text{subcarriers}}$ vectors of $f_s T_S$ samples This requires $N_{\text{subcarriers}} f_s T_S Q_{\text{bits}}$ bits for Q_{bit} quantization
Modulation of subcarriers	$N_{\text{subcarriers}}$ multiplications of subcarrier vectors by scalars $\{I_{i,k}\}_k$
Summation of subcarriers	$N_{\text{subcarriers}}$ vector sums of modulated subcarriers
Addition of Cyclic Prefixes	Append final L samples of symbol to the beginning of symbol Each symbol now requires $(f_s T_S + L)\, Q_{\text{bits}}$ bits of storage
Addition of Memory Phases	Subtract difference between final sample of symbol $(i-1)$ and initial sample of symbol i from each sample in symbol i.
Phase Modulation	Convert from samples $\nu'[n]$ to $e^{j\nu'[n]}$

Suppose we have created a new sequence $\{s'_n\}$ by adding a cyclic prefix to symbol i in the sequence $\{s_n\}$ for all i as described in the preceding paragraph. Our choice of half-sines for our subcarrier functions ensures that $q_m(t) \to 0$ for all m at the beginning *and* end of each symbol ($t \to T_S^-$ and $t \to 0^+$). Thus we can be sure that the appendation of cyclic prefixes does not make the signal phase discontinuous within symbols. Still, there are discontinuous phase jumps across thresholds *between* symbols. That is, there are large jumps in phase between $s'_{iN'-1}$ and $s'_{iN'}$. These jumps give the baseband signal high frequency components that we would like to suppress. This can be done by a judicious choice of the memory phases θ_k.

$$\theta_1 = -\arg(s_{N-L}) = -\nu((N-L)t_s) \tag{7}$$

$$\theta_k = -\arg(s_{kN-L}) + \sum_{k'=1}^{k-1} \theta_{k'} = -\nu((kN-L)t_s) + \sum_{k'=1}^{k-1} \theta_{k'} \tag{8}$$

Notice that the recursion rules in Eq. (8) depend on s_n rather than s'_n. They can be calculated and added before the cyclic prefixes are appended. One need only know the prefix length L beforehand. At the receiver side, these phases need not be subtracted because their contributions are nullified at the detector by virtue of $\int_0^{T_S} dt\; q_i(t) = 0$. Table 1 gives an account of the mathematical complexity of the digital operations that we have discussed so far.

The final step before amplification is to convert to an analogue passband signal. A DAC converts the samples $\{s'_n\}$ to a continuous waveform $s'(t)$. This wave-form can still be written as a time-dependent phase, $s'(t) = e^{j\nu'(t)}$, but because of the inclusion of cyclic prefixes $\nu'(t) \neq \nu(t)$. Regardless, the passband signal has constant amplitude when mixed with a carrier $A_c e^{2\pi j f_c t}$ at frequency f_c. If this signal is sent directly to the channel, it has the form:

$$y(t) = \mathrm{Re}\{s'(t)\, e^{2\pi j f_c t}\} = A_c \cos(2\pi f_c t + \nu'(t)) \tag{9}$$

We can model the nonlinear amplification of the signal in Eq. (9) using the Rapp model [7, 8] of solid state power amplifiers. In this model, a signal with instantaneous form $A(t)\, e^{j\phi(t)}$ is amplified resulting in a signal of the form $g[A(t)]e^{j(\phi(t)+\Phi[A(t)])}$. The non-linear gain can be written as follows.

$$g[A(t)] = \frac{A(t)}{\left(1 + A(t)^{2g}\right)^{\frac{1}{2g}}} \tag{10}$$

This amounts to a rescaling of the signal amplitude by a factor that is non-linearly dependent on the input signal amplitude. Signals with high amplitude fluctuations over time will produce in-band intermodulation products [1] that will interfere with reception. These distortions are the primary reason for choosing constant-envelope OFDM over regular OFDM. CEOFDM will have superior performance through non-linear amplification provided that the phase distortion is negligible, $\Phi[A(t)] \approx 0$. While models of solid state power amplifiers commonly set $\Phi[A(t)] \approx 0$, future work will experimentally validate this assumption.

2.2 CE-OFDM Reception

Incoherent reception necessitates added processing in the CE-OFDM receiver. However this is done in the back-end after the filtered receive signal has been sampled. Because the data vector $\{d_l\}$ is binary, each CE-OFDM symbol can assume any of $2^{N_{\text{subcarriers}}}$ possible configurations. The optimal receiver would correlate the received signal with each of these possible instantiations and detect the symbol that had the highest correlation. This requires $2^{N_{\text{subcarriers}}}$ correlation templates each with N real samples to be stored in memory, a computationally impractical task. We instead leverage the orthogonality of the subcarriers over a symbol duration to design a suboptimal correlation receiver requiring only $N_{\text{subcarriers}}$ templates of N samples. Fig. (3) shows the receive chain for such a receiver.

After filtering, the complex signal $r(t)$ is sampled at rate f_s yielding I and Q samples. Assuming fine synchronization (within $O[f_s^{-1}]$) by means of a pilot pulse (see section 2.3), we can pick the $N' \times N_{\text{symbols}}$ samples belonging to the data symbols and their prefixes. Let r_n be the n-th sample from this set with $0 \leq n \leq N' \times N_{\text{symbols}} - 1$. Knowing that the prefixes are of length L, we can delete the first L samples of each symbol and organize the remaining complex samples into an $N_{\text{symbols}} \times N$ matrix $[r_{k,m}]$ where $r_{k,m}$ is the m-th sample of the k-th symbol. The data is written into the phase of these samples. We demodulate the phase by taking $\arg(r_{k,m})$. Within symbol durations, this yields a discontinuous function of time because the range of $\arg(\bullet)$ is $[0, 2\pi)$ whereas the range of Eq. (3) is contained within $[-N_{\text{subcarriers}}A, N_{\text{subcarriers}}A]$. However, knowing that the desired phase function is continuous within symbols by construction, we can unambiguously define the "unwrapped phase" [9] as a function of time for each symbol. In perfect channel conditions, the unwrapped phase for symbol k will be equal to $v(t)$ sampled at the appropriate time steps. Let us call the i-th sample of the unwrapped phase sequence $\hat{v}_i$. It is an estimate of the samples of $v(t)$. We mentioned previously that the added memory phase θ_k is constant throughout each symbol and can be neglected as it integrates away during symbol detection.

We mitigate the effects of multipath fading by frequency domain channel equalization. As we see in Fig. (3), equalization consists of taking an N-point discrete Fourier transform (DFT), multiplying the vector element-wise by the equalization coefficients (see section 2.4), and returning to the time domain with an inverse discrete Fourier transform (IDFT). Among linear equalizers, our scheme minimizes symbol estimation errors. However it remains to be seen if linear equalization will be sufficient to reverse the effect of the multipath channel or if more advanced schemes like Decision Feedback Equalization or Turbo Equalization must be implemented to avoid the noise enhancement that all linear equalizers suffer. For theoretical calculations of error rates in the presence of white noise, see reference [2].

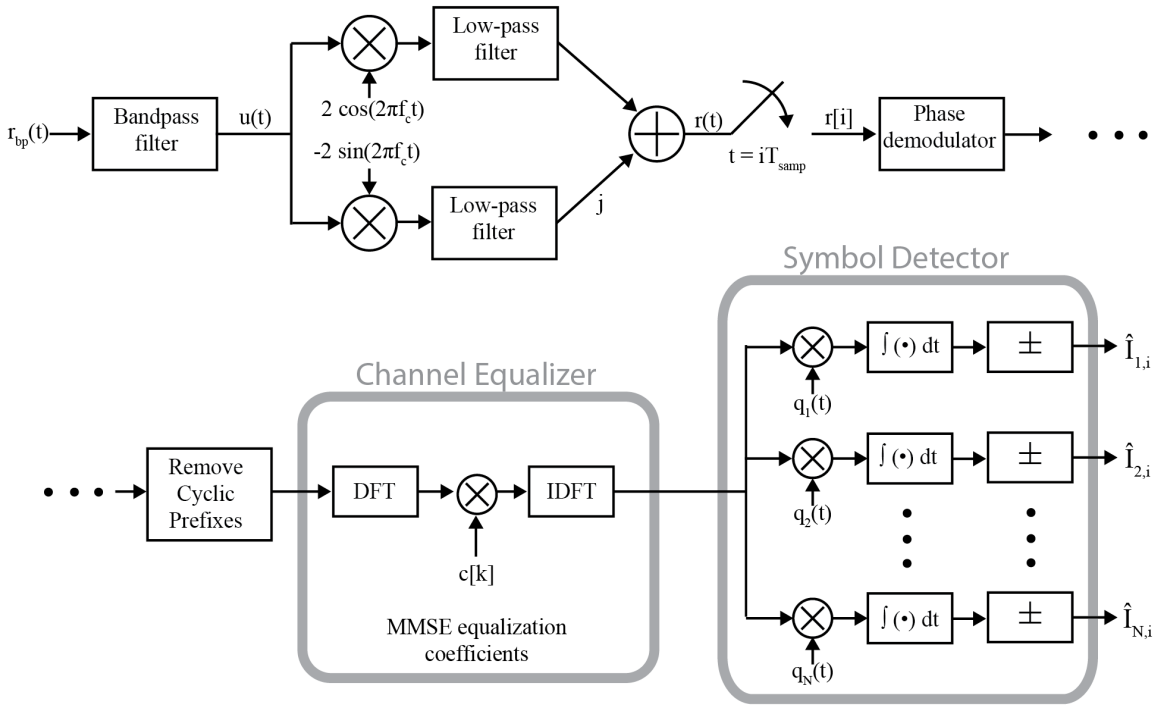

Fig. 3 — The CE-OFDM receive chain.

Now that we have an estimate $\hat{\nu}(t)$ of the phase of the received baseband signal, the next step in the chain is to estimate the matrix $[I_{i,k}]$ from Eqs. (1) and (3). Our estimate $\hat{I}_{i,k}$ is equal to 1 or -1 depending on if the correlation of symbol i with sub-carrier k is positive or negative respectively. Letting $q_{k,m} = q_k(mt_s)$ be the m-th sample of sub-carrier k,

$$\begin{aligned} \hat{I}_{i,k} &= \operatorname{sgn}\left(\sum_{n=iN}^{(i+1)N-1} \hat{\nu}_n q_{k,n \bmod iN}\right) \approx \operatorname{sgn}\left(\int_{iT_S}^{(i+1)T_S} dt\ \hat{\nu}(t)\, q_k(t)\right) \\ &\approx \operatorname{sgn}\left(\int_{iT_S}^{(i+1)T_S} dt\ \nu(t)\, q_k(t) + \sum_{m\le i} \theta_m \int_{iT_S}^{(i+1)T_S} dt\ q_k(t)\right) \end{aligned} \tag{11}$$

We see that the contribution to $\hat{I}_{i,k}$ from the memory phase vanishes by virtue of our choice of subcarriers with $\int_0^{T_S} dt\ q_k(t) = 0$. The last step in the receive chain is to reorganize $\hat{I}_{i,k}$ into an single vector estimate $\{\hat{d}_p\}$ of the transmitted data by means of a parallel-to-serial converter not shown in Fig. (3).

2.3 Frame Construction

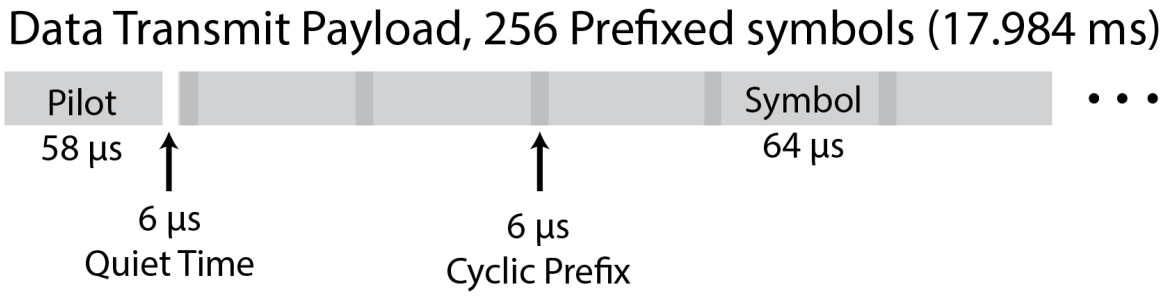

Fig. 4 — The 40 ms scheduling frame.

In Fig. (4), we see our intended communications frame broken down into its components. We are interested primarily in bursty communications because we want our sensors to primarily spend their time sensing. Thus our communications frames are designed for a single burst transmission of a packet. We are limited by the sampling rate (1 MHz) of the USRP, so our frame duration is 17.984 ms consisting of 256 symbols each with a duration of 70 μs including a 6 μs cyclic prefix. Within the total 17.984 ms, we must send a pilot signal for synchronization and channel estimation. If we were to send this data in a single stream, the symbol times would have to be on the order of 5 μs with the precise value depending on the size of the symbol constellation. While this symbol time is longer than the maximum delay time (2.5 μs) measured in one potential use-case [10], we wish to provide a migration path to more advanced hardware and higher data-rates where symbol rates of order 10 μs are required but delays of less than 1 μs would create heavy inter-symbol interference that will degrade performance. Short integration time also make the symbols more vulnerable to noise. Thus decided to use the parallelization of CE-OFDM even though it is not strictly necessary to provide the intended data-rate for our sensor network.

The calculation of the bandwidth W of our CE-OFDM waveform is a non-trivial task [2], but a conservative estimate bounds the bandwidth from below by the frequency of the highest subcarrier. This estimate is valid when $2\pi h < 1$. We presently use $2\pi h = 0.6$.

$$W \geq \frac{2N_{\text{subcarriers}}}{T_S} \tag{12}$$

Even though Eq. (12) is a lower bound, approximately 99% of the signal's power is within W of the carrier frequency [2]. Consequent to the frame construction in Fig. (4), we can conservatively estimate the bandwidth at 500 kHz. This is far from a wide-band signal, but that is because of the 1 MHz sampling rate we use with our hardware-limited radios. In future scenarios, we will either upgrade to hardware capable of supporting wider bandwidths ($\tilde{2}$0 MHz) or increase the sampling rate of our radios.

A wide bandwidth is desired so that most of the waveform is unaffected by the deepest fades of a frequency-selective channel. Although we know it to be possible, we have not yet explored how much h

and p from Eqs. (3) and (2) should be tuned to increase the frequency diversity of the waveform. Still more work must be done to ascertain if we need specialized source coding to ensure that errors have the required time correlation properties for the error correction codes to work. The choice of error correction code is an ongoing problem.

2.4 Channel Estimation, Synchronization, and Equalization

As indicated in Fig. (4), we estimate the channel impulse response using a pilot at the beginning of each 1 ms communication frame [11]. We then equalize in the frequency domain [2, 11]. Mathematically our goal is to estimate the channel frequency response $H(k)$ and then determine equalization coefficients C_k that reverse the effect of the channel as in Fig. (3). We begin with our choice of pilot signal. We presently use a Chu sequence defined as follows

$$s_n = \begin{cases} e^{j\pi rn^2/N} & \text{for even } N \\ e^{j\pi rn(n+1)/N} & \text{for odd } N \end{cases} \tag{13}$$

where N is the number of samples in the pilot symbol, and r is an integer relatively prime to N. We simply use $r = 1$. Similar to a chirp, this sequence is constant amplitude and suffers no degradation from nonlinear amplification. Its autocorrelation in time approaches a delta function with increasing sequence length. It also has flat power spectrum across the band of interest. This makes it useful for channel estimation. The total pilot symbol is 64 μs long. Within this period, 58 μs are devoted to the Chu sequence with a 6 μs quiet time afterward.

Channel estimation begins with the pilot-aided synchronization. Suppose we are assured (through shipboard synchronization) that the beginning of the pilot symbol will be received at an unknown time within a time-window $[t_0, t_0 + \delta]$. On the assumption that multipath delays are less than the length of a cyclic prefix, the last component from the pilot will be received before $t_0 + \delta + T_S'$. The goal of synchronization is to denote the integration limits in Eq. (11) that estimate $I_{i,k}$ with the fewest errors. Let the lower limit of the first symbol detection integral be t_{data}. Presently we use the autocorrelation properties of the Chu sequence to home in on the multipath component with the highest power. That is, we define

$$t_{\text{data}} = \frac{T_S'}{2} + \text{argmax}_{0 \le m \le \delta} \left\{ \sum_{n=1}^{N'} x[n] y[n+m] \right\} \tag{14}$$

where x_n is the known sampled pilot and y_n is the received sampled signal for $t \in [t_0, t_0 + \delta]$. However our frequency domain equalization by cyclic prefix relies on homing in on the *earliest* multipath component. Our present strategy probably suffices for troposcatter channels with ranges less than 100 km because the highest power multipath component is usually the earliest to be received [12]. This is not the case for longer ranges, and we may need more advanced algorithms for these ranges. Poor synchronization will result in a phase rotation of the symbol constellation after equalization. This can be corrected, but clearly more work must be done to fully explore our options. After synchronization, Fourier transforms of the received and known pilots can be compared to estimate the channel using the LMMSE criterion as in [11]. We follow this prescription to determine the estimates $\hat{H}_{LMMSE}$ before defining the coefficients in Fig. (3) to be

$$C_k = \frac{\hat{H}^*_{LMMSE}}{|\hat{H}_{LMMSE}|^2 + SNR^{-1}} \tag{15}$$

3. CE-OFDM ON GNU RADIO

GNU Radio is a software toolkit used for signal processing and wireless communications. It is highly compatible with the Universal Software Radio Peripheral (USRP) by Ettus Research and serves as the foundation for running and testing the CE-OFDM waveform in our experiments. GNU radio contains many signal processing operations useful for CE-OFDM modulation and demodulation. They include filtering, multiplication, integration, and decimation. GNU Radio also provides a template to build and create custom modules. This is especially useful given the unique properties of the CE-OFDM waveform that necessitate in-house solutions.

Most GNU Radio applications and scripts are written in Python. However the mathematical operations and signal processing are commonly performed in C++. GNU Radio blocks are first written in C++ because of the languages computational efficiency. Then through the Simplified Wrapper and Interface Generator (SWIG) interface, these C++ modules get a Python interface. The Python interface is important because the language is extremely flexible, extensible, and user-friendly. GNU Radio also contains a GUI-based programming application called GNU Radio Companion. It allows users to create programs by selecting and connecting various GNU Radio blocks in a graph.

GNU Radio uses a flow graph convention. Signal samples and data begin with a particular source, go through a chain of signal processing blocks, and finally stop at a sink. This allows flexible, modular programming for each source, sink, and block. It also enables interoperability between custom blocks and other GNU Radio blocks. This section details the GNU Radio architectures for our CE-OFDM transmitter and receiver.

3.1 Transmitter Implementation

Figure (5) depicts the CE-OFDM transmitter implementation. The entire transmitter chain is composed in Python. Recalling that our CE-OFDM implementation sends data in bursts, the data volume of a single packet was chosen to be 512 bytes. A physical layer modulation function was written to take the payload data and create baseband I/Q samples. These samples include the pilot, quiet time, and CE-OFDM signal. Once the samples were created, they were packaged as a GNU Radio message. The messages were then stored into a message queue in preparation for transmission through the flow graph layer.

The flow graph layer contains the message source and USRP sink. It performs the core real-time transmission, and runs on a separate CPU thread. The message queue is used as a common memory space for both the flow graph thread and the main thread. Once messages are inserted into the queue, the message source extracts the message and converts it into complex I/Q streams for the USRP sink. The USRP sink block accepts the baseband samples, upconverts the signal, and sends it out to the connected USRP device. The user sets the parameters for the USRP including the sample rate, transmitter center frequency, transmission gain, and antenna. These parameters are all handled by various function calls within the USRP sink. Once the block is properly connected, and the flow graph begins, the USRP radio initiates and performs the necessary RF amplification, downconversion, and decimation into baseband IQ samples.

3.2 Receiver Implementation

Figure (6) represents the block diagram implementation of the receiver. The receiver begins with a USRP Source. The USRP source performs downconversion from RF to baseband in-phase and quadrature

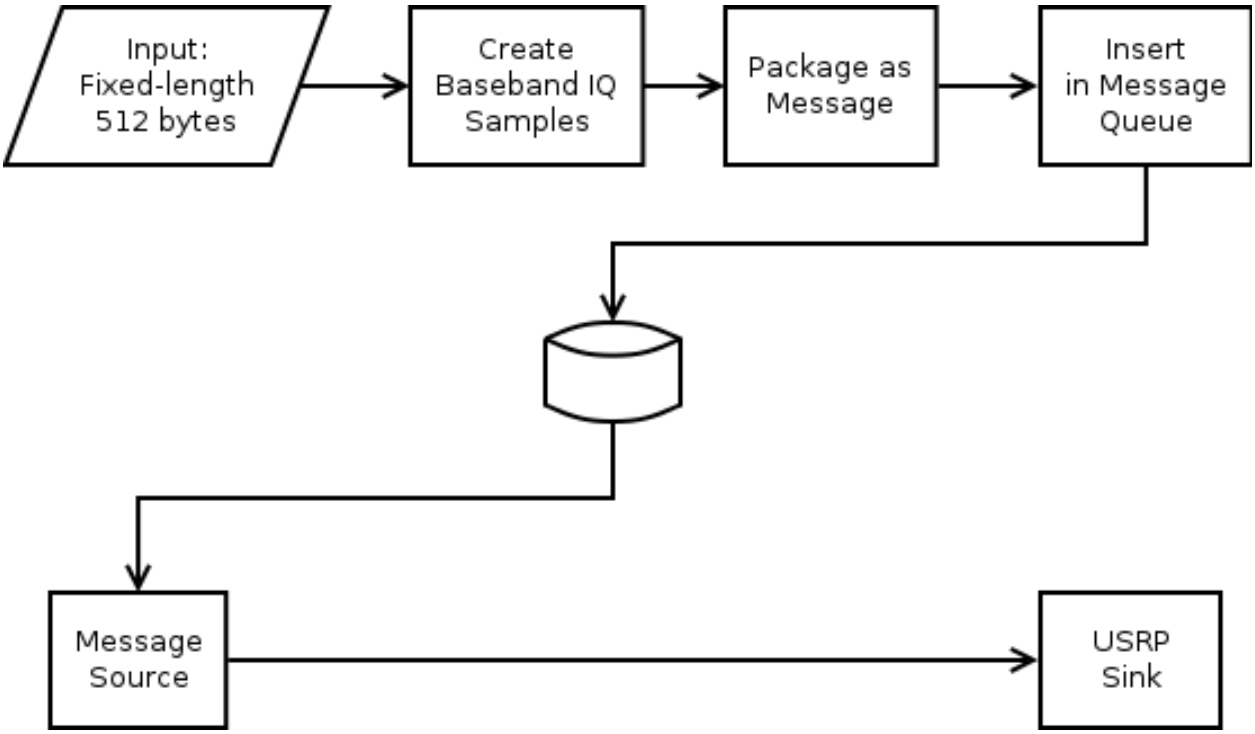

Fig. 5 — The GNU radio transmit chain.

samples. Next, the pilot detector checks for the existence of a packet. The packet valve analyzes the pilot detector and either blocks the samples or passes samples at the desired packet length. Once a full packet is passed, the channel equalization block removes any channelization effects and also discards the cyclic prefix. The phase information is extracted, using the arctangent block and unwrapping block. The symbol detector takes the phase information and correlates with all the subcarriers, determining the data bits sent. Finally, the frame sink extracts the data from the packet and stores it in a queue, available for processing by the main thread.

Two demodulation blocks were created to provide a simple interface to the CE-OFDM receiver. First, a CE-OFDM demodulator block called demod was created. It includes a pilot detector, packet valve, channel equalizer, arctangent, unwrapper, and symbol detector. The instantiation parameters are the low and high thresholds from the pilot detector. These thresholds determine the minimum correlation between the received RF signal and the expected pilot for a packet to be detected. The input stream is complex baseband I/Q samples. The first output to the block is from the symbol detector. This provides the user with a continuous and readable stream of unpacked bytes. The second output comes from the correlation output of the pilot detector. This output is particularly useful for debugging and setting the correct threshold values.

The second demodulation block is called demod_pkts. It contains the entire receiver chain, from pilot detector to framer sink. This block accepts the pilot detector thresholds as parameters to the block. For input, the block accepts complex I/Q baseband samples. It has no output because it contains a framer sink. Within the framer sink, a message queue exists to store the received packets.

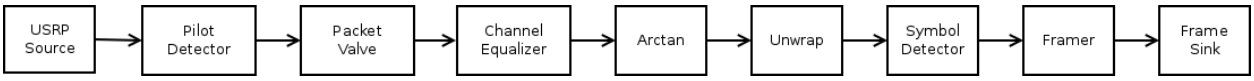

Fig. 6 — The GNU Radio receive chain for CE-OFDM.

3.2.1 Pilot Detection and Synchronization

After the USRP source block converts the RF into baseband I/Q-samples, the pilot detector attempts to find the beginning of the packet. This is performed by first correlating the signal with a reference pilot, and then thresholding the values. Figure (7) shows the GNU Radio Companion flow diagram for the pilot detector. The detector is set up as a hierarchical block i.e. a general-purpose block made out of smaller signal processing blocks that are grouped together for functional and organizational purposes. The pilot detector contains two parameters: a low threshold and a high threshold used directly by the GNU Radio threshold block. The high threshold determines the margin to set the output (the conclusion as to whether a pilot signal has been detected) to 1. Similarly, the low threshold determines where to set the output to 0. In many cases, both threshold values can be set to the same number, creating a simple slicing operation to determine the output.

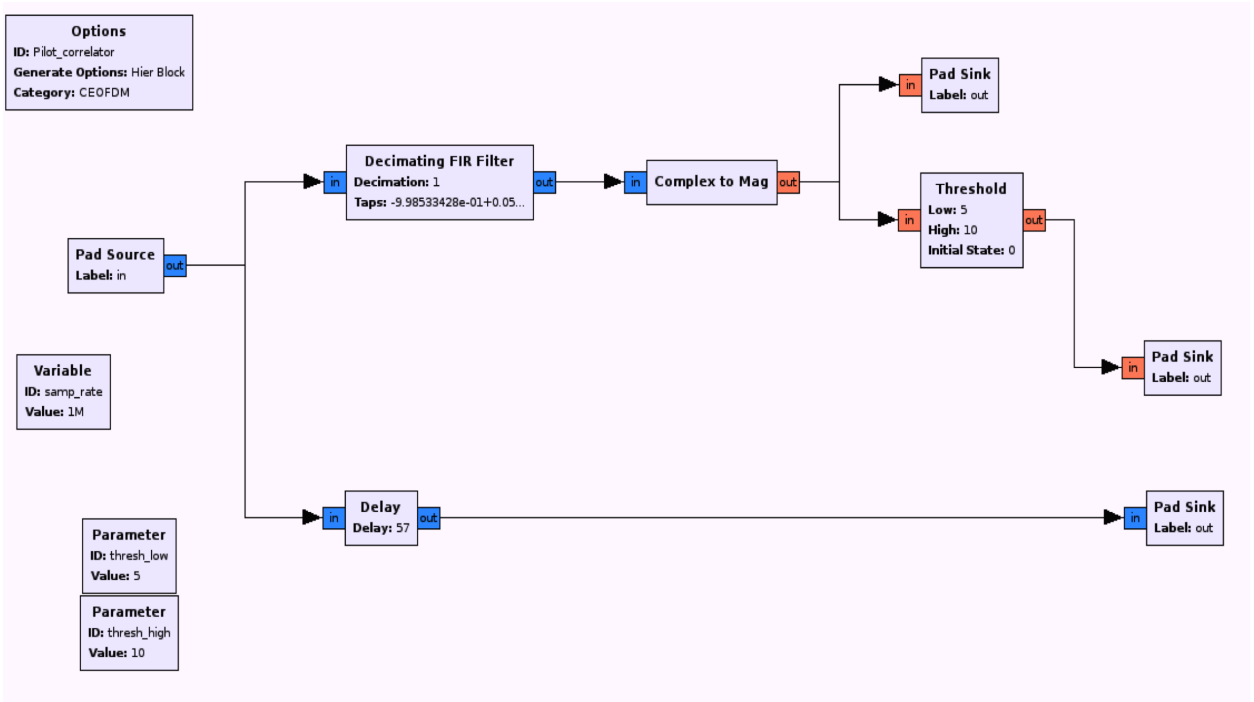

Fig. 7 — The GNU Radio Companion flow diagram for the pilot synchronizer.

The pilot detector block contains one input from the pad source. This is a stream of complex samples from some source (e.g. USRP or File). The detector correlates these against the pilot, which is a length $N = 58$ Chu sequence from Eq. (13). In general, the correlation of the pilot sequence involves taking a dot product with its conjugate.

$$C[n] = \sum_{m=n}^{m=n+N-1} R[m]P^*[m] \tag{16}$$

where $C[n]$ is the pilot correlation, N is the pilot sequence length, $R[n]$ is the baseband received samples, and $P^*[n]$ is the complex conjugate of the pilot. We use the correlation block from GNU radio to execute the operation in Eq. (16). We merely use the convolution block with the taps flipped and the output delayed by $N-1$. As Fig. (7) shows, we then run the signal through a decimating FIR filter with decimation set to 1 (indicating filtering only; no decimation). After filtering, the magnitude is output and thresholded.

The pilot detector contains three outputs as indicated by the three pad sinks in Fig. (7). The first output of the pilot detector contains the threshold output. Its values are either 0, indicating no pilot found or 1 indicating a pilot was found. The second output is a delayed copy of the input. The delay is necessary so as to use the convolution block for correlation. With the delay implemented the outputs are aligned and the beginning of the packet can be found by extracting the second output at the same index where the first output is high. The third output contains the magnitude of the correlation. This output is especially useful for debugging purposes and determining a useful threshold. In many cases, the correlation can vary greatly, depending on transmitter and receiver gain, and noise levels. This output can be used for measuring and finding an appropriate threshold in a given communication environment.

3.2.2 Packet Valve

The packet valve block controls the flow between the pilot detector and the rest of the demodulation blocks. If a pilot is detected, the packet valve engages the packet demodulation functions. It was written in C++, and it uses the GNU Radio API for out-of-tree modules. Like most common GNU Radio signal processing blocks, it has a Python interface for flow graph connections. The packet valve takes two inputs. Its first input is the correlation threshold signal from the output of the pilot detector block. The second input comes from the delayed complex samples of the pilot detector. Where the correlation threshold signal is 1, the corresponding complex sample will be at the beginning of a packet. As its output, the packet valve passes the baseband samples over the known duration of one whole packet. Once finished, the valve searches the threshold input for more potential packets. During times when no packet is found, the block does not generate any output. In these cases, the valve shuts off the output, preventing unnecessary computations downstream. By using fixed-length packets, the demodulator blocks are kept simple, and no conditional cases are needed to determine exactly when packet transmission have stopped.

3.2.3 Channel Equalizer

The channel equalizer block helps remove the effects of multipath and other channel distortions. The equalizer block is implemented in C++ using the GNU radio API, and it performs the two-fold function of cyclic prefix removal and MMSE equalization. The block works in multiple stages. First it accepts inputs from the packet valve. Since fixed length packets are used, the equalizer block can determine the first and final samples of the packet. The block then takes the received pilot and quiet time, and performs

a fast Fourier transform on them. The Fourier transform sequence is used to estimate the channel. The MMSE-based channel estimation method specifies the estimated channel coefficients $E[k]$ as follows.

$$E[k] = \frac{R[k]P^*[k]}{|P[k]|^2 + \frac{1}{\gamma}} \tag{17}$$

Where $R[k]$ is the Fourier transform of the received pilot signal and quiet time input to the equalizer block. $P[k]$ is the Fourier transform of the pilot signal including the quiet time, and γ is the estimate of the signal-to-noise ratio. The SNR is determined by using the envelope mean as the approximate signal power, and the envelope variance as the approximate noise power. With channel estimation completed, we calculate the MMSE equalization coefficients $C[k]$.

$$C[k] = \frac{E^*[k]}{|E[k]|^2 + \frac{1}{\gamma}} \tag{18}$$

The equalization block performs frequency domain equalization using these coefficients. First, the cyclic prefix of each symbol is removed. Next, the remaining samples in the symbol go through an FFT. Each symbol's Fourier components are multiplied by the equalization coefficients $C[k]$. Finally, an inverse Fourier transform is performed to obtain the equalized baseband samples. The output of the block is the cyclic prefix-removed packet I/Q sample stream, equalized via MMSE.

3.2.4 Phase Demodulation

Because CE-OFDM is a phase modulation technique, the next step is to extract the phase from the equalized I/Q samples. GNU Radio has a simple converter block called complex-to-arg that performs this operation. To properly map the phase from the domain $[0, 2\pi)$ to $\mathbb{R}$ as described in section 2.2, the phase must be unwrapped. Then the underlying phase-modulated signal can be reconstructed for symbol detection. The unwrapper block is coded in C++, and uses the phase unwrapping algorithm similar to MATLABs unwrap() function. To avoid false wraps propagating throughout the entire packet, this block unwraps one symbol at a time.

The last step is to detect symbols and convert to a received bit-stream. The symbol detector takes the unwrapped phase of the symbol, correlates with the known subcarrier signals, and makes a bit decision for each subcarrier. It is a hierarchical block, and the diagram for its operation is shown in Fig. (8). The input to the block is split 16-ways; one for each subcarrier. Because the subcarriers are mutually orthogonal, a dot product operation can be performed between the unwrapped phase and each subcarrier. This yields 16 scalar values (one for each subcarrier) per symbol. Whether or not these scalars are positive or negative determines whether a 1 or a 0 bit was received.

The unwrapped phase is input to the hierarchical symbol detection block as a single float stream. It is then converted to multiple vectors of 64 floats, each corresponding to a single symbol. Each vector is fed into 16 instances of GNU Radio's multiply-constant block where the constant parameter in the block is the corresponding subcarrier's output samples in vector format. Next, the vector output is converted back into a single floating stream before being input to the integrate block. With the decimation parameter also set to the symbol length, the integration block sums the vector producing the scalar output for each subcarrier. In Fig. (9), we find the flow graph for subcarrier correlation.

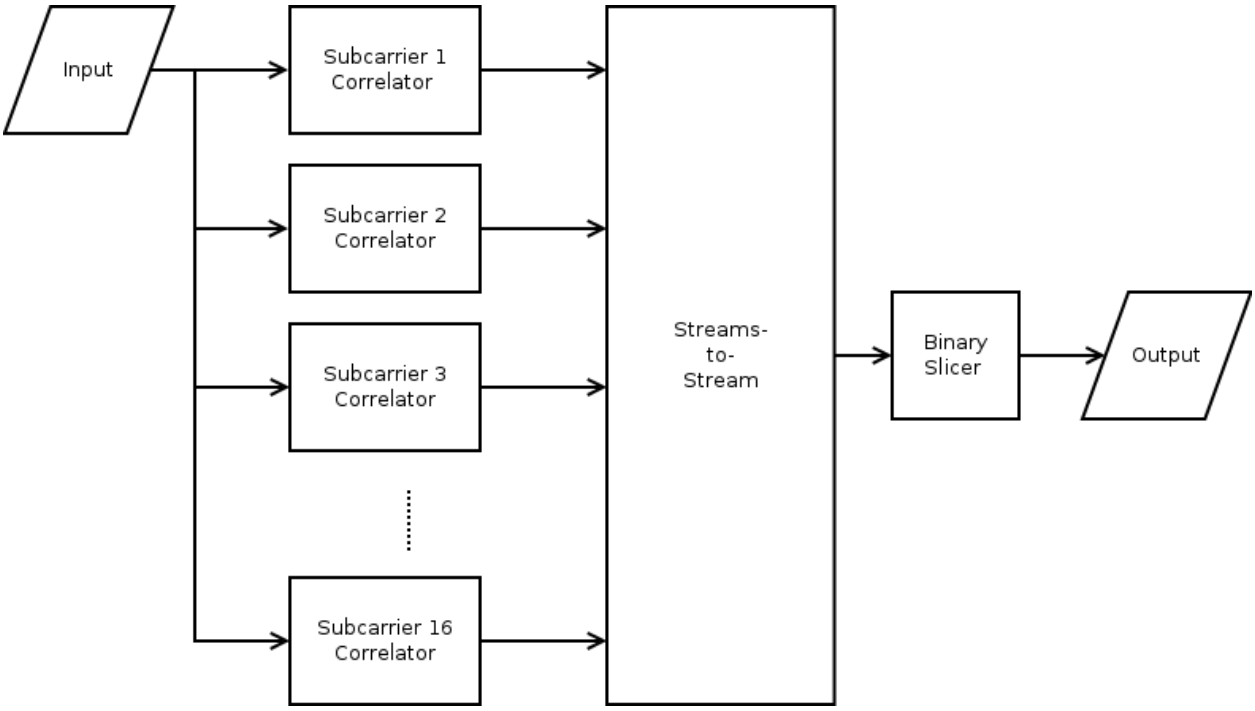

Fig. 8 — The GNU Radio symbol detection block diagram.

The output of each subcarrier correlation goes to a streams-to-stream block. This block acts as a parallel-to-serial converter multiplexing each correlator's output to a single stream. A binary slicer is then used to make the bit decision. Because each subcarrier undergoes binary modulation, the slicer simply looks at the sign of the correlation output and determines if the bit is one or zero. The output is a stream of bytes, with each byte representing one bit. Therefore the value of each byte stream is either a one or zero. Although the output is inefficient, given a maximum of 8 bits in a byte, it allows for easy viewing if testing or debugging is needed.

3.2.5 Packet Framer

The packet framer takes the stream of unpacked bytes from the symbol detector, and adds a header for the framer sink. The block is custom-made and programmed in C++. The header consists of two identical 16-bit shorts, each representing the payload size. The payload follows after the header. It is also in unpacked bytes, with the least significant bit representing a data bit. The second bit is used to mark the start of the payload. The framer sink takes the header and payload information from the framer and assembles it into a packet. The packet is stored in a message queue. Both the framer sink and the message queue are objects available in GNU Radio. They allow easy access to received packets. This is especially important given that GNU Radio scripts are usually run in a multi-threaded environment. The message queue is needed to pass received packets from the flow graphs thread to the main thread.

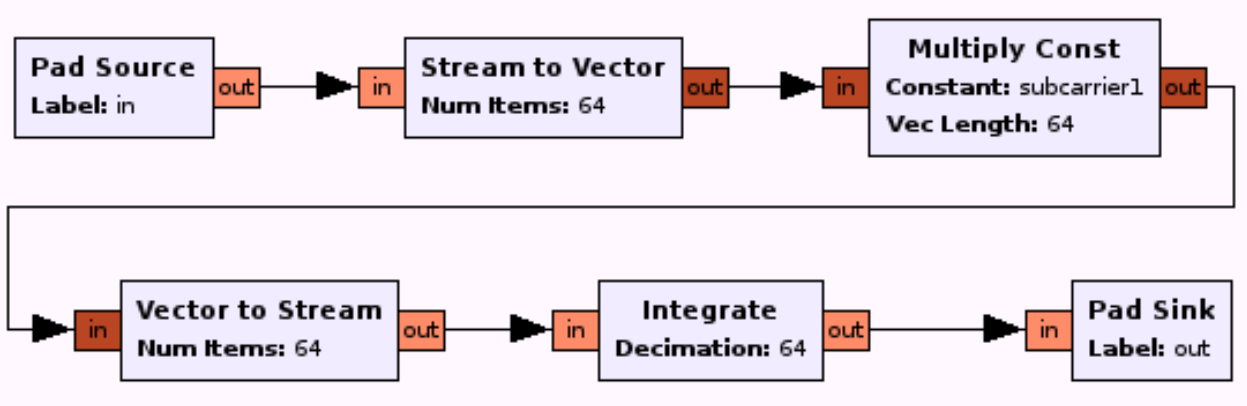

Fig. 9 — The GNU Radio Companion flow diagram for the subcarrier correlator.

4. EXPERIMENTAL AND SIMULATED RESULTS

To validate CE-OFDM as a candidate waveform for point-to-point links between distributed sensors, we constructed a bench-top experiment between two GNU radios. For added verification, we also simulated the experimental conditions in Matlab. The goal of these efforts was to produce the BER as a function of the SNR in the channel (see Fig. (10)). Both the experiments and the simulations reported here are conducted in a simple AWGN channel using the parameters in Table 2. Future validation of the waveform in complex channel conditions with Doppler and multipath spreading is on our agenda. We will conduct more robust channel studies once we acquire channel emulation capabilities.

Table 2 — Packet Parameters for CE-OFDM Experiment and Simulation

Experimental Parameter	**Parameter Value**	**Comments**
Receiver Sample Rate	1 MHz	
Center (Carrier) Frequency	900 MHz	
Bare Symbol Duration	64 μsec	
Cyclic Prefix Duration	6 μsec	This precedes each 64 μsec symbol.
Pilot Signal Duration	58 μsec	
Quiet Time Duration	6 μsec	This occurs after the pilot transmission and before the first symbol.
Number of Symbols	256	
Number of Sub-carriers	16	
Total Number of Bits	4096	
Total Packet Duration	17.984 ms	

The experiments were conducted in one of two configurations. In the first configuration, GNU radio is only used to add AWGN at a known power level, up-convert baseband samples of the waveform to 900

MHz, transmit the RF through a wired connection or over the air, and down-convert the RF on receive. The baseband signal processing, modulation, and demodulation of those samples is done on Matlab using the same functions employed in the simulation. Because the experiment and simulation use the same modulation and demodulation signal processing, we expect good agreement between the two. In fact, we find that the bit-error performance between the simulation and these experiments match reasonably in Fig. (10) for runs where equalization was not employed. However because baseband I/Q files have to be received and stored by GNU radio before being processed by Matlab offline, this communication experiment is not real-time.

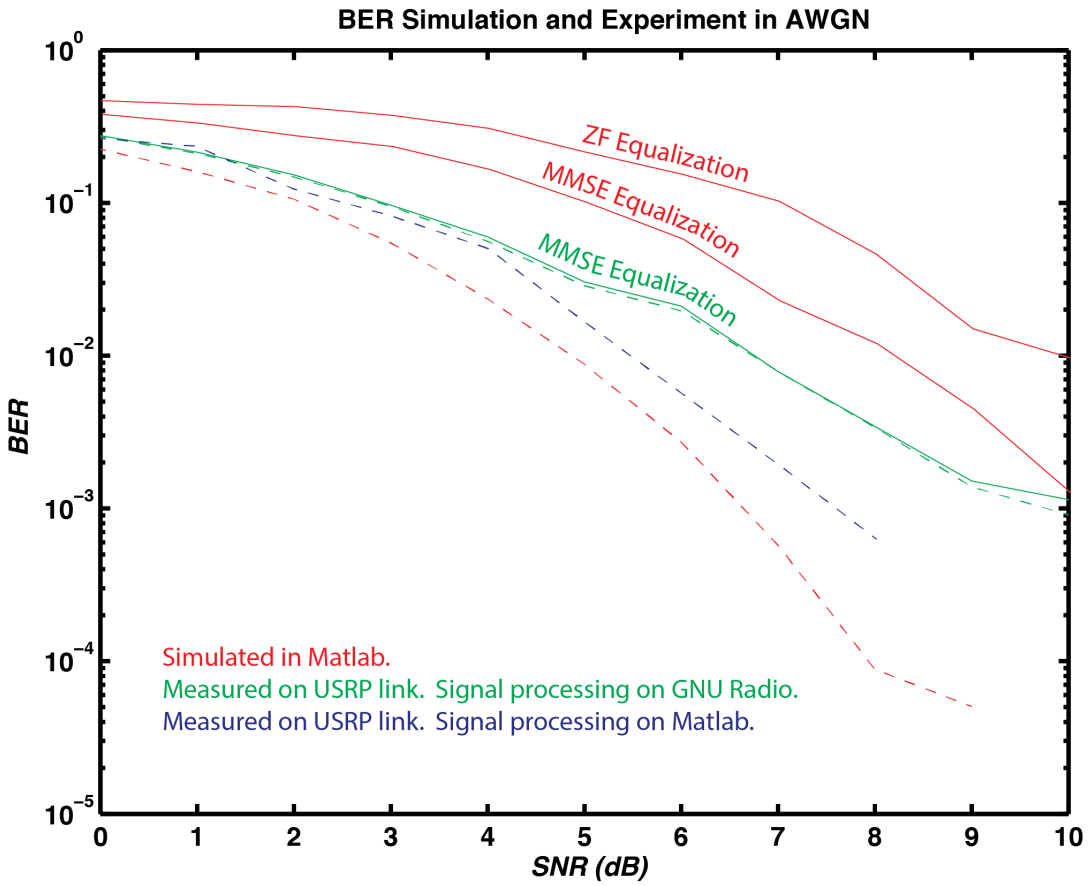

Fig. 10 — Comparison of simulated and measured BER vs. SNR with and without MMSE and ZF channel equalization. Dashed lines indicate that no equalization was used.

In the other (more recent) experimental configuration, signal processing blocks were created in C programming language to enable GNU radio to process both the baseband and RF waveform. Matlab is no longer necessary, and the system can communicate in real-time. This configuration will enable the waveform to be used by GNU radio sensors executing distributing sensing protocols, but it has yet to be fully debugged and validated. The results we discuss below indicate that there are still implementation differences between the two configurations that are leading to higher error rates in the real-time system.

Given a configuration, an experiment consisted of a series of 100 transmitted packets with identical parameters (see Table 2) at a given SNR. The communicated data was randomized between packets. The two Ettus USRP radios were connected directly by an RF cable. The AWGN was added at the transmitter using a GNU radio function in order to have precise control of the SNR. Both the simulations and the experiment enabled the option to choose channel equalization strategies. However because the channel simply consists of the addition of AWGN, we expect linear equalization to *increase* bit error rates through noise amplification with zero-forcing (ZF) performing worse than minimum mean-squared error (MMSE).

In Fig. (10) we see the average bit error results of our experiments and simulations as functions of the SNR. The most obviously feature is that the equalization strategies (represented as solid lines) do indeed fair worse than not using equalization at all (represented as dashed lines). This is because both ZF and MMSE equalizations are examples of *linear* equalization whereby the received signal is multiplied by some matrix to produce the equalized signal. Because this correction matrix always multiplies the noise in the received signal as well as the target signal, the noise is amplified. One can think of linear equalizers being useful only when channel distortions result in more errors than the addition of noise. Because there are no distortions in a pure AWGN channel, our equalizers perform worse than our unequalized waveforms both in simulation and in experiment. It was also expected that ZF would have worse performance than MMSE due to its higher noise amplification. This too is proven in the performance curves.

Another result worth pointing out is that the experiments wherein Matlab was used as the baseband signal processor outperform the real-time GNU radio implementation at higher SNR's. While the unequalized experiment with the Matlab processor usually stayed within 5 dB of the simulated results, the full GNU radio implementation only had comparable performance up to an SNR of 4 dB. The real-time implementation also differed in that no large discrepancy between equalized and unequalized strategies was exhibited. There is a possibility that some bug in the implementation is allowing the equalization routine to run even when the user specifies that it should not. Future debugging and validation will explore this possibility.

4.1 Non-Linear Amplification of CEOFDM

In order to demonstrate that the constant amplitude of our CEOFDM waveform increases the resilience to non-linearities in the amplification process, we conducted our bit error experiments through a compressed amplifier. The output of the transmitting USRP was input to a Hewlett Packard power amplifier (model number 8447D). So as not to damage the receiving USRP, the output of this amplifier was subsequently attenuated using a variable attenuator set to decrease the signal power by 61 dB. Because noise is added to the signal at the USRP transmitter, the noise is amplified along with the signal maintaining a constant SNR. Using the USRP transmit gain, we toggled the amplifier between the linear region at a USRP gain of 0 dB and a region in excess of the amplifiers 2dB compression point at a USRP gain of 30 dB. For each of these two cases, we sent a total of 2000 packets with waveform parameters given in Table 2 and recorded the number of bit errors in each received packet.

The results are plotted in Fig. (11). The average bit-error probability for the compressed and uncompressed runs were approximately equal at 2.3×10^{-4}. Furthermore, the distributions over the numbers of errors in a packet for the two cases look remarkably similar. At such a high compression, one would expect a waveform with any vulnerability to non-linear distortion to show severe increases in the probability of errors for the runs with a USRP gain of 30 dB as compared to the runs at 0 dB. That is, one would expect the red bars in Fig. (11) to be much higher than the blue bars. The absence of this effect is evidence that our CEOFDM waveform is indeed resilient to non-linear amplification as intended.

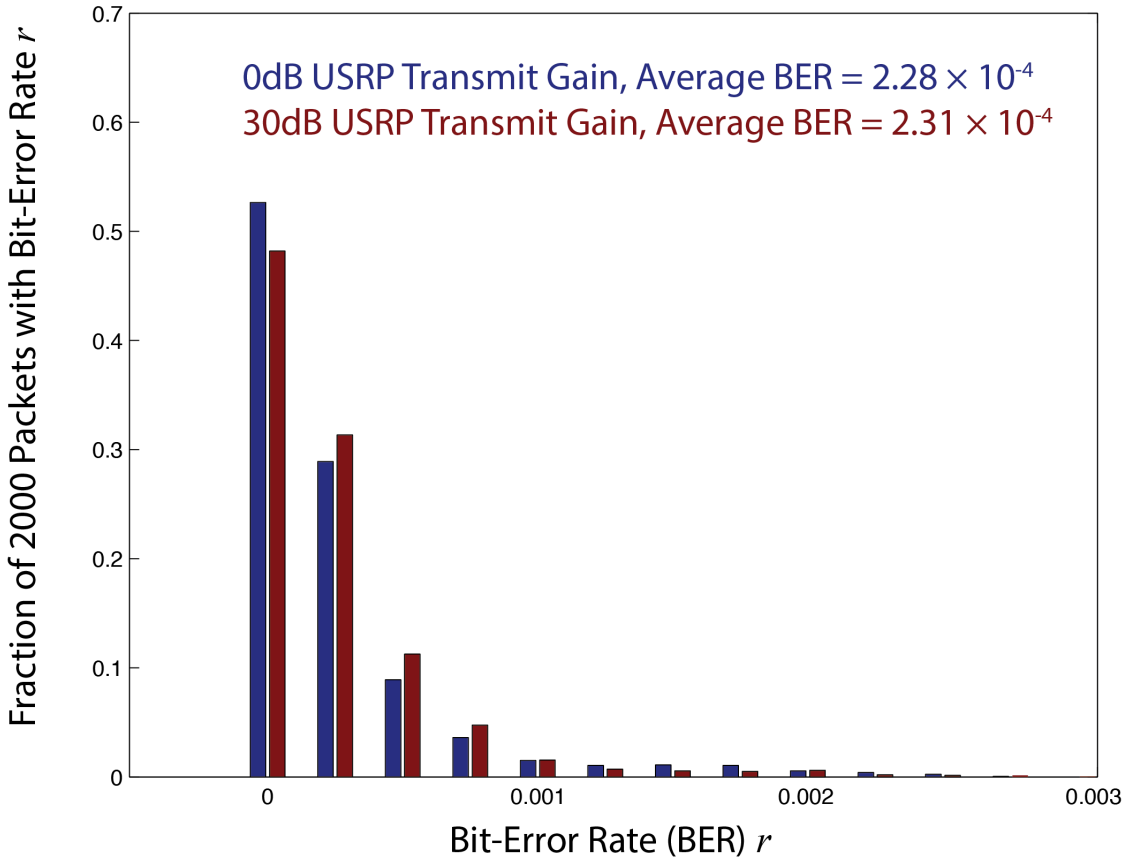

Fig. 11 — Comparison of bit error performance of CEOFDM with and without non-linear amplification.

4.2 Concluding Remarks

Constant Envelope OFDM is a wide-band, constant-amplitude waveform that can simultaneously meet demands for high data-rate and long symbol times. This makes the waveform ideal for distributed sensor networks separated by great distances. The high data rate allows sensors to communicate their data in short bursts so that they can spend most of their processing power and duty cycle on sensing their environments. The heavy parallelization of data streams (reminiscent of OFDM) permits long symbol times without compromising data rate. This offers resilience against noise and multipath. Finally a CEOFDM signal can be transmitted through a deeply compressed non-linear amplifier without in-band interference from amplifier distortions. This is due to the CEOFDM waveform's constant amplitude over time, and it makes the waveform ideal for transmission over the long distances present in highly distributed sensor networks.

We demonstrated CEOFDM between two Ettus Research USRP radios using the GNU Radio software suite. We added packet-level synchronization and frequency-domain channel equalization by means of an initial training sequence (pilot) at the beginning of each sent packet. The link was demonstrated over a

channel with a controllable level of additive white Gaussian noise, and the bit-error rate was measured as a function of the signal-to-noise ratio. These results were compared to simulations of the link across a space of different channel equalization strategies. We further demonstrated the resilience of the waveform to non-linear amplification by recording the bit-error performance of transmissions sent through a compressed amplifier. In the future we will characterize link performance through an emulated channel with nontrivial multipath, through a non-linear amplification suite, and over a troposcatter link. We will also consider new channel equalization strategies that reduce the noise amplification.

REFERENCES

1. M. O'Droma, Y. Lei, E. Bertran, and P. Gilabert, "Analysis of inter-modulation products and nonlinear distortion in RF OFDM transmitter systems," in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pp. 1–5 (IEEE, 2009).

2. S. C. Thompson, *Constant Envelope OFDM Phase Modulation*, PhD thesis (University of California, San Diego, 2005).

3. M. Geile, M. Fischer, T. O'Connel, J. Zeidler, and S. Thompson, "ARM phase I STTR final report: CE-OFDM," Contract Number N00014-05-C-0327" (January 2005).

4. C. V. Valk, "ARM phase II STTR final report: CE-OFDM," Contract Number N00014-05-C-0327" (February 2007).

5. C. V. Valk, "ARM phase II option STTR final report: CE-OFDM," Contract Number N00014-05-C-0327" (December 2008).

6. S. Signell, "OFDM systems – why cyclic prefix?," Applied Signal Processing Course Tutorial at Royal Institute of Technology" (2010).

7. C. Rapp, "Effects of HPA-nonlinearity on a 4-DPSK/OFDM-signal for a digital sound broadcasting signal," in *In ESA, Second European Conference on Satellite Communications (ECSC-2) p 179-184 (SEE N92-15210 06-32)*, volume 1, pp. 179–184 (1991).

8. E. Costa and S. Pupolin, "M-QAM-OFDM system performance in the presence of a nonlinear amplifier and phase noise," *Communications, IEEE Transactions on* **50**(3), 462–472 (Mar 2002), ISSN 0090-6778, doi: 10.1109/26.990908.

9. K. Itoh, "Analysis of the phase unwrapping algorithm," *Appl. Opt.* **21**(14), 2470–2470 (Jul 1982), doi: 10.1364/AO.21.002470, URL `http://ao.osa.org/abstract.cfm?URI=ao-21-14-2470`.

10. M. Ngo, "Troposcatter communications experiment," Internal Report ,Naval Research Lab" (January 2013).

11. A. U. Ahmed, S. C. Thompson, and J. R. Zeidler, "Channel estimation and equalization for CE-OFDM in multipath fading channels," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1–7 (IEEE, 2008).

12. P. A. Bello, "A troposcatter channel model," *Communication Technology, IEEE Transactions on* **17**(2), 130–137 (1969).