gr-owc: An Open Source GNURadio-Based Toolkit for Optical Wireless Communications

Arsalan Ahmed
Electrical and Computer Engineering department
Habib University
Karachi, Pakistan
aa03980@st.habib.edu.pk

Michael B. Rahaim*
Engineering department
University of Massachusetts, Boston
Boston, MA, USA
Michael.Rahaim@umb.edu

***Abstract*—Research in the field of optical wireless communications (OWC) has increased significantly over the past decade. As new researchers begin to explore this field, there are various hurdles related to developing a research program in OWC. This is particularly noticeable when comparing the breadth of available RF toolkits to the limited number of openly available OWC tools. Furthermore, many of the available OWC resources are standalone with limited capabilities for higher layer integration. In this work, we present *gr-owc*, an open source out-of-tree module for the widely used GNURadio signal processing toolkit. Within *gr-owc*, we have developed signal processing blocks for OWC channel simulation and common OWC modulation/demodulation techniques. We demonstrate use of this module to simulate a multi-cell/multi-user OWC network. We also describe how *gr-owc* signal processing blocks can be deployed in a physical implementation using Software Defined Radio (SDR) hardware, namely the universal software radio peripheral (USRP). Furthermore, methods for contribution to the project are described since the project is designed for continued development internally and with the goal of engaging the broader OWC research community.**

***Index Terms*—Optical Wireless Communication (OWC), Software Defined Radio (SDR), GNURadio, LiFi, Testbed**

I. Introduction

The growing demand for wireless capacity along with the advancement of laser and light emitting diode (LED) technology has led to an increasing interest in the optical medium for wireless connectivity. Accordingly, the number of active researchers in the field of optical wireless communications (OWC) is growing significantly. Early research in OWC was focused on physical layer communications and signal processing; however, the field is steadily moving towards system development, higher layer network integration, and cross-layer design [1]–[4]. Tools have been developed for simulated analysis of the optical channel and performance evaluation from a physical layer perspective [5], [6]. However, the OWC field is lacking openly available toolkits with real-time signal processing capabilities. The availability of such tools would greatly advance the field by improving researchers' ability to evaluate performance at higher layers and in the context of intelligent/adaptive systems.

With this in mind, we look to the RF community which has benefited greatly from the concept of software defined radio (SDR). In particular, the open-source GNURadio signal processing toolkit and similar toolkits (e.g., Matlab/Simulink and LabView) have had an incredible impact on the field of RF communications [7], [8]. When extending the SDR concept to OWC, we consider three key objectives:

- **Core Library:** The toolkit should provide common signal processing blocks, sample signal processing chains, and shared resources. This core functionality will accelerate the startup process while allowing researchers to work with existing implementations and benchmark designs.
- **Modularity:** Users should be able to define unique signal processing chains or interchange components of a signal processing chain following modular design principles. In addition, it should be easy for users to move between simulated and physical systems or swap between hardware implementations. This modularity allows for iterative design changes and enables the use of "golden references" to evaluate improvements to known techniques.
- **Structure:** The underlying structure that allows data to flow through the signal processing chain should be unnoticeable to the user, allowing them to focus on fundamental research rather than instantiation challenges. The availability of an imperceptible coding structure reduces development time and streamlines the process of testing and verification for new signal processing techniques.

Considering these objectives, we present *gr-owc*, an open source OWC signal processing toolkit that is developed as an out-of-tree (OOT) module in the GNURadio framework. The rest of the paper is organized as follows: Section II provides background information related to OWC and SDR. Section III defines theoretical models for the OWC channel and OWC-specific modulation schemes. Section IV describes the software framework and integration with GNURadio while Section V describes specific signal processing blocks developed for *gr-owc*. Section VI introduces sample signal processing chains that exemplify the use of *gr-owc* in practice. Section VII concludes the paper.

II. Background

Over the past two decades, SDR has gone from a novel concept to a key enabler of configurable RF communication systems. Current work in SDR often highlights the functional ability to implement dynamic and adaptive systems. However, the ability to quickly move ideas from theory or simulation to

978-1-6654-2854-5/21/$31.00 ©2021 IEEE

practice is another significant aspect of the "software-defined" concept. This design flexibility has been incredibly valuable to the research community. By reducing the barrier to entry and making it more feasible to physically instantiate novel ideas, SDR has created a more equitable opportunity for advanced research in the field of wireless communications.

OWC channel simulators have offered valuable insight into physical layer performance; however, there is significant effort required to move from physical layer analysis to performance analysis in multi-cell/multi-user networks. Some researchers have developed tools that integrate OWC physical layer models within network simulators [9]–[11]; however the move from simulated network analysis to proof-of-concept implementation is not feasible without real-time signal processing tools. To the best of the authors' knowledge, the only openly available real time signal processing library for OWC is the OpenVLC project [12], [13]. This project offers an excellent low cost research tool for visible light communication (VLC) research. However, the limiting throughput of 400kb/s makes it impractical for more advanced research in high speed OWC networks. The use of *gr-owc* and moderately priced USRP equipment enables a middle ground between the available low rate open source tools and proprietary high rate OWC components. Furthermore, the ability to implement signal processing on a general purpose processor and use USRPs to integrate with custom optical front-end hardware allows researchers to adapt the signal throughput to the characteristics of whatever hardware they have available. This can range from very inexpensive commercially available components with bandwidth on the order of 1MHz to recently commercialized front-end hardware designed specifically for OWC, such as the LiFi R&D kit from Hyperion Technologies.

The use of SDR tools for deploying OWC systems is not completely novel. We have demonstrated "software defined visible light communication" links in prior work [14]–[17]. Similarly, other research groups have utilized SDR equipment to implement novel OWC designs [18]–[20]. However, this prior work has primarily been developed for specific research objectives and the underlying source code is not available in an easily accessible open-source format. The *gr-owc* library is fully accessible under the GPLv3 license, integrates directly with GNURadio, and offers both user-plane and developer-plane functionality. User-plane functionality allows researchers to utilize the existing library in order to create custom signal processing chains and parameterize individual blocks. The developer plane allows researchers to create custom signal processing blocks to incorporate within their signal chains and, potentially, add these blocks to the *gr-owc* library.

III. Optical Wireless Communication

The use of optical signals implies a unique directionality that enables incredibly high area capacity (i.e., b/s/m^2); however, optical signals must account for potential occlusions due to obstructions in the line-of-sight (LOS) path and signal loss for highly mobile devices. Furthermore, the use of intensity modulation with direct detection (IM/DD) adds unique requirements such that the transmitted signal must be real-valued and non-negative. Furthermore, average optical power constraints on many OWC systems differ from the conventional electrical power constraints of RF systems [21]. Moreover, some OWC systems define average optical power requirements rather than limits. This is particularly relevant to dual-use VLC systems that are used for both illumination and data transfer. In such systems, the average optical power must be specified, and potentially controllable, in order to meet illumination and dimming requirements of the lighting system. Accordingly, the unique characteristics of OWC imply a need for OWC-specific modulation schemes and channel models.

A. OWC Modulation and Coding

Given the use of IM/DD for OWC, the modulated time domain signals must be real-valued and non-negative. This often implies the use of baseband pulsed modulation schemes such as on-off keying (OOK), pulse-amplitude modulation (PAM), and pulse-position modulation (PPM). In order to address the illumination requirements and dimming characteristics of dual-use VLC systems, variable pulse-position modulation (VPPM) was introduced in the IEEE 802.15.7 standard such that the data value is represented by the pulse position and the light intensity is controlled by the duty cycle [22]. Similarly, various modifications to conventional orthogonal frequency division multiplexing (OFDM) have been introduced to apply OFDM to the optical channel [23]. These techniques include DC-biased optical OFDM (DCO-OFDM), asymmetrically clipped optical OFDM (ACO-OFDM), and multiple variants that provide high spectral efficiency and dimming control [24]–[26].

B. OWC Channel

The LOS path is typically dominant in OWC; therefore, the LOS path's DC gain is a good approximation of an OWC link's DC gain and the multipath component can be ignored in most cases where a LOS path exists. The DC channel gain from transmitter i to receiver j is generally defined as:

$$H_{ij} = \frac{G_T\left(\phi_{ij}\right) G_R\left(\psi_{ij}\right)}{d_{ij}^2} \tag{1}$$

where the channel parameters ϕ_{ij}, ψ_{ij}, and d_{ij} represent the emission angle, acceptance angle, and distance between transmitter and receiver. $G_T\left(\right)$ and $G_R\left(\right)$ define the transmit and receive gain as functions of the emission and reception angles, respectively. These functions are defined below for the commonly used point source model with Lambertian emission.

$$G_T\left(\phi\right) = \frac{m+1}{2\pi}\cos^m\left(\phi\right) \tag{2}$$

$$G_R\left(\psi\right) = AT_s g\left(\psi\right)\cos\left(\psi\right) \tag{3}$$

For this model, $G_T\left(\right)$ is parameterized by the Lambertian order, m, and $G_R\left(\right)$ is parameterized by the photosensor area, A, the optical filter transmittance, T_s, and a gain function, $g\left(\right)$, for a concentrator lens with refractive index n and field-of-view (FOV) Ψ_C. The concentrator gain is defined as $g\left(\psi\right) = n/\sin^2\left(\Psi_C\right)$ for $0 \leq \psi \leq \Psi_C$ and 0 otherwise.

In a system with multiple transmitters and receivers, the total optical signal power incident on a given receiver's photosensor can be defined as the sum of the received power from all transmitters. The conversion between electrical and optical domains must also be accounted for when modeling an SDR OWC implementation. For simplicity, we assume the transmitted electrical signal and received optical signal are maintained within the linear conversion range of the front-end hardware. In this case, the electrical to optical conversion factor, C_T [W/A or W/V], and the optical to electrical conversion factor, C_R [A/W or V/W], can be defined as constants such that the transmitted and received electrical signals from a given transmitter are related by the constant factor $C_T C_R H_{ij}$. As such, the received signal at receiver j is defined as:

$$y(t) = \sum_i C_T^{(i)} C_R^{(j)} H_{ij} x(t) \tag{4}$$

While ignoring the optical channel's multipath component simplifies the frequency response in the optical domain to a flat spectral response, the frequency characteristics of the optical conversion do have a significant impact on the transmitted signal. These characteristics can be modeled by observing the transmitter, optical channel, and receiver as three systems in series. This is demonstrated in Section VI.

IV. Code Development

The *gr-owc* library is developed within the GNURadio framework. GNURadio is an openly available software development toolkit that enables the implementation of custom SDR signal processing chains (i.e., flow-graphs). It provides a core library of available signal processing blocks and supports the integration of custom flow-graphs with various types of hardware for physical testing. One of GNURadio's biggest advantages is the ability to extend its functionality beyond what is offered by the core library. In particular, the development of OOT modules enables the addition of custom signal processing blocks. We use this feature to develop a set of OWC-specific blocks within our custom *gr-owc* library. The blocks within *gr-owc* can be easily combined with signal processing blocks from the core library and integrated with SDR hardware.

A. GNURadio

Since GNURadio is the basis of *gr-owc*, this section serves as a quick-start for those who do not have any prior experience with the GNURadio toolkit. At the most abstract level, GNURadio views a signal processing application as a signal chain consisting of individual processing stages, or blocks, that are connected together so that data can flow from block to block. A GNURadio **block** provides a specific functionality according to which it processes the incoming data stream and/or generates an outgoing data stream. Blocks are connected using virtual wires and can be parameterized according to the application's requirements. A signal chain consisting of connected blocks is termed as a **flow-graph**.

GNURadio flow-graphs are generated within the Python programming language by essentially defining blocks, block parameters, and the virtual connections between blocks. Once a flow-graph is defined, the GNURadio framework manages the passing of data signals between blocks without requiring attention from the user. This structure is one of the fundamental benefits of GNURadio. For simplicity, a graphical interface called **GNU Radio Companion (GRC)** is packaged within GNURadio. In GRC, users can build visual flow-graphs using drag and drop functionality. GRC then auto-generates the underlying Python code to execute the flow-graph.

A GNURadio **module** contains a group of related blocks and GNURadio's core library contains a comprehensive list of modules that can be used to develop custom flow-graphs. Custom modules can be developed as well. These custom modules reside outside GNURadio's main source tree and thus, are named **Out of Tree (OOT) modules**. Once installed, blocks from the OOT modules hold no distinction when compared to the core GNURadio blocks and both can be used in the same flow-graph. When creating OOT modules, users define the module characteristics, the processing capabilities of blocks within the custom module, and testing/documentation for custom blocks. Custom blocks can be written in Python or C++. We have developed the *gr-owc* blocks using C++ while Python is utilized for writing the unit tests for these blocks.

B. User Plane: Installation and Use

There are several ways to install GNURadio. We suggest the Python Build Overlay Managed Bundle System (PyBOMBS). PyBOMBS is efficient in building GNURadio with OOT modules and allows the GNURadio installation to exist in specified directories instead of system folders. The latter is particularly useful as it allows users to have multiple local installations (i.e., prefixes) giving them the liberty to specify a prefix for deploying the OOT module. Detailed instructions for installing GNURadio via PyBOMBS can be found on the PyBOMBS GitHub page [27]. Once GNURadio is installed, *gr-owc* can be installed in a GNURadio prefix. Detailed instructions and specific installation commands can be found at the *gr-owc* GitHub page [28]. After installing *gr-owc*, the example flow-graphs from the GitHub repository can be run to verify the installation and the blocks from *gr-owc* can be used in custom flow-graphs.

C. Developer Plane: Module / Block Creation

The current *gr-owc* release has a set of functional blocks described in Section V; however, the authors aim to continually develop this library and incorporate contributions from the broader OWC research community. GNURadio's *gr_modtool* is used to create the structural code for new blocks and add them to an OOT module. Once a block is created, there are five files that need to be edited: The .py file is used to write Quality Assurance (QA) test(s) that are run when building the module. The .cc file specifies the type of input and output, the number of incoming and outgoing samples, and the work function that defines the actual signal processing. The two .h files need to be edited if the block requires callback functions that allow

parameters to be edited while the flowgraph is running. Finally, the .yml file specifies how the blocks are displayed in GRC.

V. Signal Processing Blocks for *gr-owc*

Currently, the *gr-owc* module includes blocks for OWC-specific IM/DD channel modeling along with modulator and demodulator blocks for various pulsed modulation schemes. This section describes the blocks in the initial release and Section VI reviews example flow-graphs that use these blocks.

A. Channel Models

The *gr-owc* channel blocks implement the DC channel gain of a point source Lambertian emitter and optical receiver as described in Section III. The blocks offer the flexibility to define the optical channel gain or, alternatively, the electrical channel gain if it is assumed that (a) the transmitter and receiver operate in the linear conversion range and (b) the conversion factors are known. Separate blocks are provided with parameters for either the relative or absolute position/orientation of the transmitter/receiver pair(s).

- **OWC Channel (relative):** This block relates the electrical signals from N transmitters to received electrical signals at M receivers for a given set of emission/acceptance angles and distances for the NM transmitter/receiver pairs. Lambertian orders are parameterized for the N transmitters. Photosensor area, optical filter gain, refractive index and concentrator FOV are parameterized for the M receivers. Conversion factors are specified for all N transmitters and M receivers. If conversion factors are set to 1, the block relates the transmitted optical power to the received optical power at the photosensor surface.
- **OWC Channel (absolute):** This block functions exactly like the channel model block described above. The difference is that it uses the absolute location/orientation of transmitters and receivers. Therefore, the block parameters include the 3D location coordinates and 3D orientation vectors of the N transmitters and M receivers. Emission angles, acceptance angles and distances are calculated directly from these absolute locations/orientations.

B. Modulators and Demodulators

The current *gr-owc* release includes a set of common OWC pulsed modulation schemes with parameters to adapt the modulator/demodulator for a given system. OOK and PAM modulators have two variants, one that takes min/max signal values and another that takes the amplitude and mean value.

- **OOK:** The modulator generates a 2-level OOK waveform from an input bit stream for the given min/max values (or amplitude and average values) and the desired number of samples per symbol. The demodulator compares the average of a set of incoming values to an input threshold and defines the binary outcome. The threshold and number of samples per symbol are parameterized.
- **PAM:** The modulator generates the multi-level PAM waveform from an input stream of decimal values. Modulation order, min/max values (or amplitude and average values), and the samples per symbol are parameterized. The *binary to decimal mapper* block can be used with the modulator to map a bit stream into decimal (i.e., symbol) values. The demodulator averages a set of real-valued input and compares this average to a set of thresholds determined from defined min/max signal values. Modulation order, min/max signal value, and samples per symbol are parameterized. The *decimal to binary mapper* block is used to map the decimal values to a bit stream.
- **VPPM:** The modulator generates a 2-level VPPM waveform from an input bit stream for the given min/max values, the samples per symbol, and number of high samples per pulse (i.e., duty cycle). The demodulator uses a matched filter to compare a received signal to the two VPPM symbol waveforms and determines the output bits according to the most comparable waveform. The samples per symbol, duty cycle, and gain are parameterized.

VI. Results and Analysis

In order to exemplify the use of *gr-owc*, we describe three example flow-graphs. The GRC files for these flow-graphs are included in the *gr-owc* repository.

A. Channel Model

We first demonstrate the *OWC Channel (relative)* block in three scenarios where Rx2 has various locations and orientations (Fig. 1). Sinusoidal signals with frequencies of 10kHz and 1kHz are passed into the channel block, modeling signals from Tx1 and Tx2, respectively. Blocks from the GNURadio core library are used to add noise and plot the time/frequency domain received signals along with a waterfall representation.

In Fig. 2, we show the time domain and waterfall plots of the three scenarios. For easier viewing, the time domain plot shows results of a slightly modified flow-graph where Rx2(a), Rx2(b), and Rx2(c) are all defined as separate receivers in the channel block. In scenario 1, the transmitter/receiver pairs are perfectly aligned and separated; allowing for the near-perfect reception with minimal interference. In scenario 2, Rx2 moves away from Tx2 and the 1kHz signal component is reduced due to the increased distance and emission/acceptance angles. The 10kHz signal from Tx1 can now be seen. In scenario 3, Rx2 is

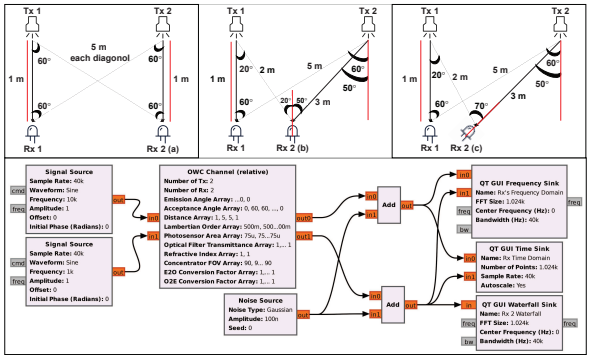


Fig. 1: Example channel simulation scenarios (*top*) evaluated with the OWC channel's sample flowgraph (*bottom*).

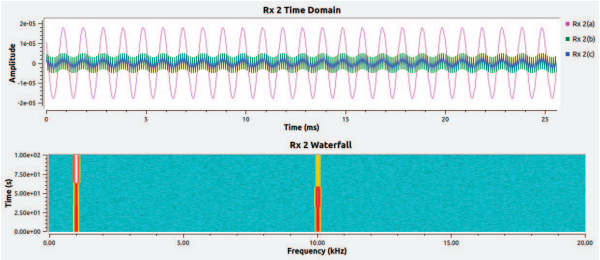


Fig. 2: Time domain results of a fixed three-receiver scenario (*top*) and waterfall results showing the relative received signal power for a dynamic scenario (*bottom*).

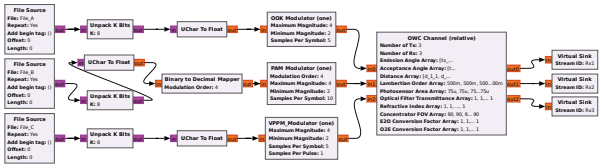
rotated towards Tx2 such that the acceptance angle is brought back to 0°. This reduces the signal from Tx1 and increases the desired signal from Tx2. Transmitter/receiver position and orientation can be adjusted dynamically while the flow-graph is running. This can be done via user control through GNURadio's available graphical widgets or by adjusting parameters at specific times in the python code to create a repeatable script. The waterfall plot in Fig. 2 demonstrates this ability by executing a flow-graph where the channel block's Rx2 parameters are changed to model the three positions at times 0s, 33s, and 66s. These examples demonstrate the ability to model the impact of (a) location and orientation; (b) interference in multi-cell/multi-user simulations; and (c) mobile scenarios that include device translational and rotational motion.

B. Modulation and Demodulation

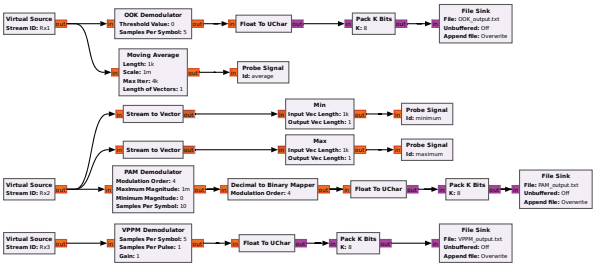
Here, we demonstrate the use of *gr-owc* modulator and demodulator blocks along with the channel block to build a simulated OWC system. Fig. 3 shows an example flow-graph with three separated end-to-end links. At each of the corresponding Tx-Rx pairs, there is a different pulsed modulation scheme (i.e., OOK, PAM and VPPM). For the OOK and PAM demodulators, receive side decoding parameters are calculated using core library blocks (i.e., mean, min and max values). Data from a text file is transmitted through each of the links and the perfect reception is observed in this noise-free scenario. This highlights the use of *gr-owc* blocks to build a comprehensive flow-graph simulating an OWC system. Modifying the receiver locations in the transmitter block can easily simulate interference scenarios and/or scenarios where multiple cells utilize joint transmission. GNURadio bit error rate calculation blocks can be utilized to simulate performance in various noise and interference scenarios.

C. DCO-OFDM

In this final example, we demonstrate how GNURadio core library blocks can be used with *gr-owc* blocks to implement a more complex OWC system. We also describe how these simulated system flow-graphs can be easily adapted to test real-time OWC transmission in a physical instantiation. The flow-graph in Fig. 4 implements DCO-OFDM using the core library OFDM transmitter/receiver blocks and custom hierarchical blocks to (a) generate the real valued DCO-OFDM



(a) *Transmit Signal Chain*



(b) *Receive Signal Chain*

Fig. 3: Signal chains for flowgraphs simulating file transfer and using *gr-owc* modulators, demodulators, and channel model.

signal for OWC; (b) simulate the DC channel gain, noise, and transmitter frequency response of the OWC link; and (c) implement a noise reduction low-pass filter (LPF) and down conversion so that the received DCO-OFDM signal is compatible with the conventional OFDM receiver block.

In the up-conversion block, the complex valued input signal is multiplied by a complex cosine with frequency well below the flow-graph sample rate and above the highest active carrier of the OFDM transmitter. This allows the real component of the complex signal to be isolated while maintaining the signal characteristics. In the OWC channel hierarchical block, the *gr-owc* channel block is combined with an additive noise block and a block simulating the frequency response of the optical transmitter. The frequency response is simulated using two consecutive filters (low pass and high pass) whose taps are tuned to produce a filter representing the characteristics of our optical front-end hardware. Finally, the down conversion with LPF hierarchical block reverses the effect of the up conversion and uses the LPF to eliminate undesirable higher frequency images stemming from the up-conversion and down-conversion process. Using this end-to-end OWC link simulation, data is sent and received successfully.

This example can be used to demonstrate the simplicity of moving from a simulated system to a real-time physical instantiation using SDR hardware and front-end OWC equipment. Specifically, this flowgraph has been implemented directly on a physical link by simply creating two separate flowgraphs for the transmit/receive signal chains and replacing the OWC channel hierarchical block with USRP sink and source blocks at the transmitting and receiving devices, respectively. The intent of this paper is not to describe the physical OWC hardware implementation and we can not fully describe the hardware configuration due to page limitations. Rather, we highlight this process as a valuable aspect of integrating the *gr-owc* library within GNURadio and note that the modular

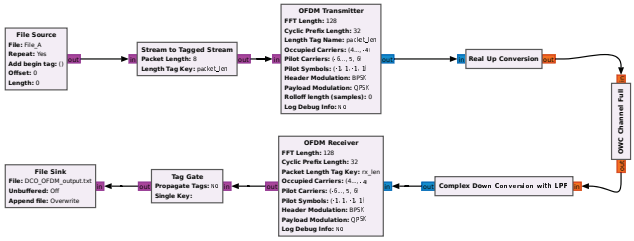


Fig. 4: Flow-graph of a simulated DCO-OFDM link with hierarchical blocks from *gr-owc* to implement up/down conversion and OWC channel simulation.

design allows the flow-graph parameters to be configured to align with the available optical front-end hardware. Multiple instances of prior software defined VLC hardware setups can be found in [14]–[16]. This simple hardware integration allows for a straight forward design process where the channel model is used for simulated analysis and parameter selection before moving over to the hardware implementation.

VII. Conclusions

We have introduced *gr-owc*, an open source toolkit for optical wireless communications. The module's signal processing blocks have been demonstrated in simulation and use of the module in a physical system has been discussed. For students and researchers who are interested in OWC, this toolkit offers a starting point for developing and analyzing OWC systems. Furthermore, OWC can be explored within the context of higher layer protocols and heterogeneous systems by integrating *gr-owc* blocks with modules in the core GNURadio libraries and/or other OOT modules. This accelerates the startup process and offers fast development / testing of novel signal processing techniques, real time adaptation capabilities, shared resources for the community to build off of, and a common analysis framework. The module is openly available [28] and continued development of the toolkit is expected with future contributions including synchronization blocks for the pulsed modulation schemes, modulators and demodulators for various optical OFDM techniques, and expansion of the channel models to include non-linear conversions, channel occlusions, propagation delays, and OWC-specific noise models.

References

[1] M.Z. Chowdhury, M. Shahjalal, M. Hasan, Y.M. Jang, et al. The role of OWC technologies in 5G/6G and IoT solutions: Prospects, directions, and challenges. *Applied Sciences*, 9(20):4367, 2019.

[2] S.U. Rehman, S. Ullah, P.H.J. Chong, S. Yongchareon, and D. Komosny. Visible light communication: A system perspective-overview and challenges. *Sensors*, 19(5):1153, 2019.

[3] M.Z. Chowdhury, M.K. Hasan, M. Shahjalal, M.T. Hossan, and Y.M. Jang. Optical wireless hybrid networks: Trends, opportunities, challenges, and research directions. *IEEE Communications Surveys & Tutorials*, 22(2):930–966, 2020.

[4] X. Wu, M.D. Soltani, L. Zhou, M. Safari, and H. Haas. Hybrid LiFi and WiFi networks: A survey. *IEEE Comms Surveys & Tutorials*, 2021.

[5] J.B. Carruthers and P. Kannan. Iterative site-based modeling for wireless infrared channels. *IEEE Transactions on Antennas and Propagation*, 50(5):759–765, 2002.

[6] Z. Ghassemlooy, W. Popoola, and S. Rajbhandari. *Optical wireless communications: system and channel modelling with Matlab®*. CRC press, 2019.

[7] D. Valerio. Open source software-defined radio: A survey on GNUradio and its applications. *Forschungszentrum Telekommunikation Wien, Vienna, Technical Report FTW-TR-2008-002*, 2008.

[8] T.W. Rondeau, O. Holland, H. Bogucka, and A. Medeisis. On the GNU radio ecosystem. *Opportunistic Spectrum Sharing and White Space Access: The Practical Reality*, pages 25–48, 2015.

[9] M. Bilgi and M. Yuksel. Packet-based simulation for optical wireless communication. In *2010 17th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)*, pages 1–6, 2010.

[10] D. Pfefferkorn, K. Helmholdt, and H. Blume. Performance estimation of indoor optical wireless communication systems using OMNeT++. In *International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–5, 2017.

[11] A. Aldalbahi, M.B. Rahaim, A. Khreishah, M. Ayyash, and T.D.C. Little. Visible light communication module: An open source extension to the ns3 network simulator with real system validation. *IEEE Access*, 5:22144–22158, 2017.

[12] Q. Wang, D. Giustiniano, and D. Puccinelli. OpenVLC: Software-defined visible light embedded networks. In *Proceedings of the 1st ACM MobiCom Workshop on Visible Light Communication Systems*, VLCS '14, page 1520, New York, NY, USA, 2014. ACM.

[13] A. Galisteo, D. Juara, and D. Giustiniano. Research in visible light communication systems with OpenVLC1.3. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 539–544, 2019.

[14] M.B. Rahaim, A. Mirvakili, S. Ray, M. Hella, V.J. Koomson, and T.D.C. Little. Software defined visible light communication. In *Wireless Communications Technologies and Software Defined Radio (SDR-WInnComm), Wireless Innovations Forum Conference on*, 2014.

[15] A. Mirvakili, V.J. Koomson, M.B. Rahaim, H. Elgala, and T.D.C. Little. Wireless access test-bed through visible light and dimming compatible OFDM. In *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, pages 2268–2272, March 2015.

[16] M.B. Rahaim and T.D.C. Little. *Visible Light Communications: Theory and Applications*, chapter Optical Small Cells, RF/VLC Heterogeneous Networks, and Software Defined VLC. Opticwise, 2017.

[17] T.D.C. Little, M.B. Rahaim, I. Abdalla, E. Lam, R. Mcallister, and A.M. Vegni. A multi-cell lighting testbed for VLC and VLP. In *2018 Global LIFI Congress (GLC)*, pages 1–6, 2018.

[18] Y. Qiao, H. Haas, and E. Knightly. A software-defined visible light communications system with warp. In *1st ACM Workshop on Visible Light Communication Systems*, 2014.

[19] W. Hussain, H.F. Ugurdag, and M. Uysal. Software defined VLC system: Implementation and performance evaluation. In *International Workshop on Optical Wireless Comms (IWOW)*, pages 117–121. IEEE, 2015.

[20] B. Aly, M. Elamassie, B. Kebapci, and M. Uysal. Experimental evaluation of a software defined visible light communication system. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[21] M.B. Rahaim and T.D.C. Little. Reconciling approaches to snr analysis in optical wireless communications. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2017.

[22] S. Rajagopal, R.D. Roberts, and S.K. Lim. IEEE 802.15.7 visible light communication: modulation schemes and dimming support. *Communications Magazine, IEEE*, 50(3):72–82, 3 2012.

[23] S.D. Dissanayake and J. Armstrong. Comparison of ACO-OFDM, DCO-OFDM and ADO-OFDM in IM/DD systems. *Journal of lightwave technology*, 31(7):1063–1072, 2013.

[24] H. Elgala and T.D.C. Little. Reverse polarity optical-OFDM (RPO-OFDM): dimming compatible OFDM for gigabit VLC links. *Optics express*, 21(20):24288–24299, 2013.

[25] Z. Wang, T. Mao, and Q. Wang. Optical OFDM for visible light communications. In *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1190–1194, 2017.

[26] X. Zhang, Z. Babar, P. Petropoulos, H. Haas, and L. Hanzo. The evolution of optical OFDM. *IEEE Communications Surveys Tutorials*, pages 1–1, 2021.

[27] GNURadio. PyBOMBS (python build overlay managed bundle system). Github, https://github.com/gnuradio/pybombs, June 2021. [Online].

[28] A Ahmed and M.B. Rahaim. UCaNLabUMB/gr-owc: Initial release. Github, https://github.com/UCaNLabUMB/gr-owc, 2021. doi: 10.5281/zenodo.5237300.