WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Design and Experimental Evaluation Framework for A Low-Cost ESP32-Based Li-Fi System for Secure Multimedia Transmission

Likith Kumar K, Rakshak G Raju, Tharunya C A

Department of Electronics and Communication Engineering
Coorg Institute of Technology, Ponnampet, Karnataka, India

ABSTRACT

Light Fidelity (Li-Fi) is an optical wireless communication approach in which information is conveyed through controlled changes in light intensity. This paper describes a low-cost prototype framework built around two ESP32 development boards, an LED transmitter, a photodiode-based receiver, and a computer application for file handling. The transmitter converts data supplied by a computer into a timed optical signal, while the receiver detects the changing light and returns the recovered data to a second computer interface. The proposed software design supports packet framing, sequence numbers, error checking, acknowledgements, retransmission, and optional authenticated encryption. The same architecture can be used to investigate the transfer of text, images, audio, video, PDF documents, and other files, provided that the optical link offers enough reliable throughput for the selected file or media bitrate. The paper follows a staged test plan covering electrical bring-up, binary transmission, distance and alignment, ambient-light conditions, and file-integrity checks. No measured bit-error rate or throughput results are claimed because a complete laboratory dataset was not supplied. Instead, the manuscript identifies the measurements required to evaluate the prototype fairly and discusses hardware limits that may affect performance.

Keywords: Li-Fi; visible light communication; ESP32; optical wireless communication; photodiode; multimedia transmission; CRC; packet retransmission

1. INTRODUCTION

Wireless data is commonly carried over radio-frequency links, but radio is not the only possible medium for transferring information. Optical wireless communication uses light to carry a signal through free space. In visible light communication (VLC), the source is generally within the visible spectrum; Li-Fi is often used to describe a more complete light-based networking approach. Haas and co-authors discuss Li-Fi as a networked form of optical wireless communication, rather than simply switching an LED on and off [1].

A basic optical link has two main parts. At the transmitting end, an LED is driven so that its light intensity changes according to the data. At the receiving end, a photodetector converts the received light into an electrical signal, which is then conditioned and interpreted. This method can be attractive in environments where optical links are useful, but its performance depends on the LED driver, receiver circuit, line of sight, alignment, distance, and background illumination [2], [5], [6].

The aim of this project is to build a practical, low-cost platform for learning how optical transmission, embedded timing, and computer-side file handling work together. The proposed setup uses ESP32 boards to manage the time-sensitive hardware tasks and Python software on a computer to prepare files, divide data into packets, monitor transfer progress, and reconstruct received files. The design is intended as an educational prototype, not as a claim of compliance with a commercial Li-Fi standard. The key contribution is a structured implementation and test plan that can be followed before making claims about transmission speed, range, or reliability.

2. LITERATURE SURVEY

2.1 Li-Fi and optical wireless communication

Haas et al. describe Li-Fi as a networked optical wireless technology and explain how it differs from a basic visible-light link [1]. Their work provides the conceptual basis for treating the LED, photodetector, and communication protocol as parts of a complete system. For this project, the distinction is important because a working LED link alone does not establish that networking functions, reliability, or multimedia performance have been achieved.

2.2 Standards for short-range optical links

IEEE 802.15.7 specifies physical- and medium-access-control considerations for short-range optical wireless communication, including factors such as ambient-light interference and mobility [2]. IEEE 802.11bb later introduced light communication into the IEEE 802.11 family [3]. These standards are useful references for understanding the field, but the custom ESP32 prototype described here does not implement their complete PHY and MAC requirements and should not be described as standards-compliant.

2.3 VLC surveys and multimedia prototypes

Jain, Sharma, and Sharma discuss visible light communication as an alternative communication approach [5], while Bao and co-authors survey Li-Fi concepts and research directions [6]. Prototype work has also explored data transmission using visible light, including multimedia-oriented demonstrations [7]. Such studies support the feasibility of investigating optical data transfer with a small prototype. However, a demonstration that transmits a file does not by itself establish a high sustained bitrate; the actual link must be measured under clearly reported conditions.

2.4 Receiver design and implementation gap

Receiver performance depends on photodetector characteristics and the analogue front end used to convert photocurrent into a usable voltage. A photodiode with a general-purpose amplifier may be suitable for an initial low-speed demonstration, but it can limit the rate at which symbols can be detected reliably [8], [11], [12]. The ESP32 analogue-to-digital converter also requires attention to input range and calibration [14]. These component-level constraints motivate a staged evaluation rather than assuming that the controller's clock frequency represents the optical data rate.

Research gap addressed by this project: the proposed work combines a low-cost ESP32 optical link with a computer-side file-transfer workflow and a repeatable evaluation plan. It emphasizes measurable reliability and file integrity instead of reporting unverified speed or distance figures.

3. METHODOLOGY

3.1 System overview

The system is divided into a transmitting computer, an ESP32 transmitter, an LED driver, a free-space optical path, a photodiode receiver circuit, an ESP32 receiver, and a receiving computer. The computer application handles file selection and higher-level data processing. The ESP32 transmitter generates the optical waveform, while the receiver samples and decodes the signal before forwarding recovered bytes to the computer.

Transmitting PC (Python) → USB Serial → ESP32 TX → LED Driver → LED ~ light path ~ Photodiode → Amplifier → ESP32 RX → USB Serial → Receiving PC (Python)

Fig. 1. Functional block diagram of the proposed Li-Fi prototype.

3.2 Hardware components

ESP32 transmitter board — Receives packet data from the computer and generates a timed optical output.

High-power LED and suitable driver — Produces modulated light; the driver must support the required switching rate and LED current.

MOSFET or dedicated gate driver — Switches LED current. The selected device must be suitable for the ESP32 logic voltage.

Power supply and heat sink — Supplies the LED driver within its rated limits and helps manage LED temperature.

BPW34 silicon PIN photodiode — Detects received visible or near-infrared optical power.

Photodiode amplifier / transimpedance amplifier — Converts photodiode current into a voltage that can be sampled.

ESP32 receiver board — Samples the conditioned signal, decodes the received data, and forwards it to the computer.

USB cables and wiring — Provide serial communication and power where appropriate; a stable common reference is required for connected circuits.

The reference circuit uses GPIO18 on the transmitter for optical output and GPIO34 on a classic ESP32 receiver for analogue input. Confirm the pin capabilities of the exact development board being used before wiring. Do not connect an LED directly to a GPIO pin or allow the photodiode amplifier output to exceed the receiver board's permitted input range.

3.3 Software components

Arduino IDE and the ESP32 board package for firmware development.

Python for file selection, packet formation, serial communication, progress reporting, reconstruction, and logging.

PySerial for communication between the computer and each ESP32 board.

CRC-based error detection and acknowledgement/retransmission logic for link reliability.

Python cryptography tools for optional authenticated encryption; keys should not be embedded in publicly shared code.

FFmpeg for optional audio/video probing, conversion, or bitrate reduction, and VLC for playback of reconstructed media.

Optional OpenCV and numerical tools for image/video processing or signal analysis.

3.4 Data-transfer procedure

Start both ESP32 boards and initialize the selected serial settings and GPIO pins.

Check the LED driver electrically at a low switching rate before attempting data transmission.

Select a text message or file on the transmitting computer. If necessary, prepare a media file using a compatible codec or lower bitrate.

Optionally encrypt the file at the computer layer, then divide the data into packets.

Each packet carries framing information, a packet type, a sequence number, payload length, payload, and an error-checking value.

The transmitter ESP32 converts incoming packet bytes into the chosen optical modulation pattern, such as on-off keying (OOK).

The photodiode and analogue front end convert changes in received light into a voltage. The receiver ESP32 samples and decodes the signal.

The receiver checks packet framing and error detection. It sends an acknowledgement for a valid packet or requests retransmission when a packet is missing or corrupted.

The receiving computer reassembles the payload, decrypts it if encryption was used, and writes the recovered file.

Compare the original and received file using a cryptographic hash such as SHA-256. Open or play the file only after the integrity check has completed.

3.5 Packet structure and reliability

A suitable packet layout is PREAMBLE | VERSION | TYPE | SEQUENCE | LENGTH | PAYLOAD | CRC. The preamble helps the receiver identify the start of a packet; the version field supports future changes; the type field distinguishes data and control messages; the sequence number helps identify missing or duplicated packets; the length field specifies the payload size; and the CRC detects accidental corruption. CRC detects errors but is not a security mechanism. If confidentiality and protection against deliberate modification are required, authenticated encryption should be applied separately at the computer layer.

The receiver should acknowledge correctly received packets. If an acknowledgement is not received within a configured timeout, the sender can retransmit the packet up to a defined limit. The software should log failed packets and stop with a clear error if the retry limit is reached, rather than silently producing an incomplete file.

3.6 Testing and evaluation plan

Testing should proceed in stages so that faults can be isolated. First, test the LED driver and receiver amplifier separately. Next, perform a wired loopback or low-rate binary test, then introduce the optical path. After the binary link is stable, test text, a small image, a short audio file, and a low-bitrate video in that order. Use at least three repeated trials for each condition and record the hardware configuration.

Distance and alignment: test selected distances and angular offsets while keeping other variables as constant as possible.

Ambient illumination: repeat measurements in a dark room, ordinary indoor lighting, and stronger external illumination.

Binary reliability: transmit a known bit sequence and record the number of erroneous bits and packets.

File integrity: compare the original and received file hashes and record pass/fail.

Transfer performance: record file size, elapsed time, effective throughput, latency, retransmission count, and packet loss.

Media validation: record file format, codec, resolution, frame rate, and target bitrate for audio/video tests.

4. RESULTS AND DISCUSSION

The supplied research material describes the prototype architecture and an evaluation procedure, but it does not provide a complete set of measured laboratory results. For this reason, this draft does not invent a maximum range, bit-error rate, throughput, latency, or multimedia success rate. The result

of the present paper is therefore a reproducible design and validation framework; quantitative performance conclusions must be added after the hardware is tested.

The proposed design separates the computer software from the optical link. This makes it possible to check packet creation, file reconstruction, and integrity verification through a software or wired loopback before investigating the optical channel. Once the optical path is active, the most important variables to record are the received waveform, distance, alignment, ambient illumination, symbol rate, packet errors, retransmissions, and effective throughput.

Multimedia support should be reported carefully. A program may accept an audio or video file, but successful selection or packetization does not prove that the optical channel can carry the file at its original bitrate. If the measured goodput is lower than the media bitrate, the transfer may need to be slowed, buffered, compressed, or transcoded. Any claim of successful transmission should be supported by a completed transfer and a matching file-integrity check.

Table 1. Recommended performance record

Parameter	**Unit / format**	**Value to enter after testing**
Optical distance	m	Measured value
Angular offset	degrees	Measured value
Symbol rate	symbols/s	Configured value
Bit-error rate (BER)	ratio	Measured value
Packet loss	%	Measured value
Retransmissions	packets	Measured count
Effective goodput	bit/s	Measured value
Transfer latency	ms	Measured value
Ambient illumination	lux	Measured value
File integrity	pass/fail	Hash comparison

5. ADVANTAGES

The prototype uses widely available development boards and components, making it suitable for an educational laboratory project.

The architecture separates PC-side file processing from time-sensitive optical modulation.

Packet framing, acknowledgements, and retransmission can improve recovery from accidental transmission errors.

File hashes provide a straightforward way to check whether the reconstructed file matches the original.

The design can be extended to test text, images, audio, video, and other binary files.

The modular structure allows the LED driver, photodiode amplifier, and software protocol to be evaluated independently.

6. LIMITATIONS

The practical data rate is limited by the LED driver, photodetector front end, sampling method, and embedded timing—not simply by the ESP32 clock frequency.

A general-purpose LM358 amplifier may restrict receiver bandwidth at higher symbol rates; a suitable transimpedance amplifier may be needed.

An IRF540N MOSFET should not be assumed to switch fully from a 3.3 V ESP32 GPIO. Use a properly specified logic-level MOSFET or a suitable gate driver.

Strong ambient light, poor alignment, increased distance, and optical blockage can reduce signal quality.

The ADC input must be conditioned to remain within the board's safe voltage range; calibration may be needed for meaningful measurements.

No quantitative experimental results are included in the current draft, so performance claims remain unverified.

The custom protocol is not interoperable with IEEE 802.15.7 or IEEE 802.11bb equipment unless the required standard mechanisms are implemented.

7. CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

This paper presents a low-cost ESP32-based Li-Fi/VLC prototype framework for investigating secure file and multimedia transfer through an optical link. The architecture combines an LED transmitter, a photodiode receiver, embedded signal handling, and computer software for packetization and file reconstruction. Reliability features such as sequence numbering, CRC checking, acknowledgements, retransmission, and file-hash verification provide a structured way to detect transmission failures. The paper also explains why the actual performance must be measured rather than inferred from component specifications. Since a full experimental dataset was not supplied, the work should be treated as a design and evaluation framework until the recommended tests have been completed.

7.2 Future scope

Replace the baseline receiver amplifier with a transimpedance amplifier selected for the target bandwidth and photodiode capacitance.

Use a MOSFET and gate-drive arrangement specified for the ESP32's actual logic voltage.

Add optical filtering, shielding, and automatic threshold or gain control to improve operation under changing ambient light.

Implement hardware-timed modulation and sampling to improve timing consistency.

Measure BER, packet loss, goodput, and latency across repeated trials and publish the raw test conditions.

Investigate forward-error correction or alternative modulation methods after the basic link is reliable.

Evaluate the transfer of progressively larger files and media with documented codecs, bitrates, and hash verification.

REFERENCES

[1] H. Haas, L. Yin, Y. Wang, and C. Chen, “What is LiFi?,” Journal of Lightwave Technology, vol. 34, no. 6, pp. 1533–1544, 2016, doi: 10.1109/JLT.2015.2510021.

[2] IEEE, IEEE Standard for Local and Metropolitan Area Networks—Part 15.7: Short-Range Optical Wireless Communications, IEEE Std 802.15.7-2018, 2019.

[3] IEEE, “IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—LAN/MAN—Part 11: Wireless LAN MAC and PHY Specifications—Amendment 6: Light Communications,” IEEE Std 802.11bb-2023, 2023, doi: 10.1109/IEEESTD.2024.10315104.

[4] IEEE, IEEE Standard for Multi-Gigabit per Second Optical Wireless Communications (OWC), with Ranges up to 200 m, for Both Stationary and Mobile Devices, IEEE Std 802.15.13-2023, 2023.

[5] A. Jain, A. K. Sharma, and S. K. Sharma, “Visible light communication: A smart way towards wireless communication,” in Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, doi: 10.1109/ICACCI.2014.6968262.

[6] X. Bao, G. Yu, J. Dai, and X. Zhu, “Li-Fi: Light fidelity—a survey,” Wireless Networks, vol. 21, pp. 1879–1889, 2015, doi: 10.1007/s11276-015-0889-0.

[7] “Complete Data Transmission using Li-Fi Technology with Visible Light Communication,” in Proceedings of the 2022 International Conference on Futuristic Technologies (INCOFT), 2022, doi: 10.1109/INCOFT55651.2022.10094453.

[8] “Review of Visible Light Communication Receivers: Standardization, Modulation Techniques, Receiver Technologies, and Application Scenarios,” IEEE Access, 2026, doi: 10.1109/ACCESS.2026.3690088.

[9] A. A. Purwita et al., “Model and methodology to characterize phosphor-based white LED visible light communication links,” Sensors, vol. 23, no. 10, Art. no. 4637, 2023.

[10] “A Comprehensive Survey on MIMO Visible Light Communication: Current Research, Machine Learning and Future Trends,” Sensors, vol. 23, no. 2, Art. no. 739, 2023.

[11] Vishay Semiconductors, “BPW34, BPW34S Silicon PIN Photodiode,” Datasheet, Document No. 81521, Rev. 2.1.

[12] Texas Instruments, “LM358 Data Sheet: Industry-Standard Dual Operational Amplifiers,” Rev. AB.

[13] Infineon Technologies, “IRF540N: 100 V Single N-Channel Power MOSFET,” product and datasheet information.

[14] Espressif Systems, “ESP32 ADC Programming Guide,” ESP-IDF Documentation.

[15] Texas Instruments, “TIDA-00978 Reference Design for Extending the OPA857 Transimpedance Bandwidth,” Reference Design.

[16] pySerial, “pySerial Documentation,” version 3.5.

[17] FFmpeg Developers, “FFmpeg Documentation,” FFmpeg Project Documentation.

[18] Python Cryptographic Authority, “cryptography Documentation,” Python cryptography library documentation.