A

MINI PROJECT REPORT

ON

LI-FI Based Data Transfer

Submitted By

Soham Rajaram Mane	**T400550142**
Parth Sachin Jadhav	**T400550125**
Siddhi Deepak Kolhe	**T400550135**

Guided By

Prof. Sushma Bhosle

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

NMVPM's and PCET's

NUTAN MAHARASHTRA INSTITUTE OF ENGINEERING & TECHNOLOGY

TALEGAON DABHADE, PUNE MAHARASHTRA 410507

2024 - 2025

LI-FI Based Data Transfer

A Mini Project Report Submitted

In Partial Fulfilment of the Requirements

For the Degree of

Bachelor of Engineering

in

Electronics & Telecommunication Engineering

Submitted By

Soham Rajaram Mane	**T400550142**
Parth Sachin Jadhav	**T400550125**
Siddhi Deepak Kolhe	**T400550135**

Guided By

Prof. Sushma Bhosle

NMVPM's and PCET's

NUTAN MAHARASHTRA INSTITUTE OF ENGINEERING & TECHNOLOGY

TALEGAON DABHADE, PUNE MAHARASHTRA 410507

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled "**Li-Fi Based Data Transfer System**" has been carried out **by Soham Rajaram Mane, Parth Sachin Jadhav, Siddhi Deepak Kolhe** under my guidance in partial fulfillment of the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering of Nutan Maharashtra Institute of Engineering & Technology affiliated to Savitribai Phule Pune University, Pune, during the academic year 2024-2025. To the best of my knowledge and belief this work has not been submitted elsewhere for the award of any other degree.

Prof. Sushma Bhosle	**Dr. Ashwini Shinde**	**Dr. S.N. Sapali**
Guide	**HOD E&TC**	**Principal**

External Examiner

Date: Place: Talegaon Dabhade, Pune-410507

NMVPM's and PCET's

NUTAN MAHARASHTRA INSTITUTE OF ENGINEERING & TECHNOLOGY

TALEGAON DABHADE, PUNE MAHARASHTRA 410507

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled "**Li-Fi Based Data Transfer System**" has been carried out **by Soham Rajaram Mane, Parth Sachin Jadhav, Siddhi Deepak Kolhe** under my guidance in partial fulfillment of the degree of Bachelor of Engineering in Electronics and Telecommunication Engineering of Nutan Maharashtra Institute of Engineering & Technology affiliated to Savitribai Phule Pune University, Pune, during the academic year 2024-2025. To the best of my knowledge and belief this work has not been submitted elsewhere for the award of any other degree.

Prof. Sushma Bhosle	**Dr. Ashwini Shinde**	**Dr. S.N. Sapali**
Guide	**HOD E&TC**	**Principal**

External Examiner

Date: 9/5/25

Place: Talegaon Dabhade, Pune-410507

ACKNOWLEDGMENT

I would like to express my sincere gratitude to all those who provided their valuable guidance, support, and encouragement throughout the successful completion of this mini project, titled "Li-Fi Based Data Transfer System".

First and foremost, I am profoundly thankful to **Prof. Sushma Bhosle**, for their continuous mentorship, insightful feedback, and unwavering support throughout the project development lifecycle. Their expert advice and constructive criticism helped me refine both the technical and research components of this work.

I am equally grateful to the Head of Department, **Dr. Ashwini Sinde**, and the faculty members of the Department of Electronics and Telecommunication Engineering, Nutan Maharashtra Institute of Engineering & Technology (NMIET), for creating an intellectually stimulating environment and providing all necessary facilities to carry out this project.

I extend my heartfelt appreciation to my colleagues and project teammates for their collaboration, creative input, and mutual encouragement during various stages of project design, implementation, and testing.

Finally, I sincerely acknowledge the constant support and motivation of my family and friends, who inspired me to strive for excellence throughout this endeavor.

This project has been a significant learning experience, and I am committed to leveraging the knowledge gained here to drive future innovations and professional growth.

I look forward to building upon this foundation to explore more advanced optical wireless communication technologies in the future.

Soham Rajaram Mane

Parth Sachin Jadhav

Siddhi Deepak Kolhe

ABSTRACT

Li-Fi (Light Fidelity) is an emerging wireless communication technology that uses visible light to transmit data at high speeds. This report presents an academic prototype Li-Fi system developed using ESP32 microcontrollers, LED, solar panels, the Arduino IDE, and Python scripting. The system transmits textual data via visible light pulses encoded in 8-bit binary, with a simple Code Division Multiple Access **(CDMA)** scheme to address specific receivers. The receiver uses a solar panel as a photodetector, converting received light intensities to voltage levels, which are quantized into binary digits. A custom frame structure with start/end markers and an 8-bit receiver code ensures that only the intended device decodes the message. The prototype successfully demonstrates one-way communication of text messages through light, with the decoded output verified on the receiver side. Key performance aspects such as bitstream timing, synchronization, and error rates were observed, and the results provide insight into the challenges and potential of visible light communication on a small scale. Finally, we discuss system behaviour, trade-offs (such as transmission lag due to low data rate), advantages, limitations, and propose future improvements including advanced encoding (Manchester coding, Base64 encoding, hamming error correction), multi-colour LED channels, Acknowledgement from the receiver and implementing two-way communication and integration with mobile devices to enhance data integrity and throughput.

ABBREVIATIONS

A. ABBREVIATIONS

Abbreviation	Full Form	First Used in Page No.
LIFI	Light Fidelity	14
CDMA	Code Division Multiplexing	10
RF	Radio Frequency	11
VLC	Visible Light Communication	12
Tx	Transmitter	26
Rx	Receiver	26

TABLE INDEX

FIGURE INDEX

INDEX

1. Introduction

Wireless communication through visible light, known as **Li-Fi**, has gained attention as a complementary technology to Wi-Fi, offering the prospect of high-speed data transfer using light from LEDs. In a Li-Fi system, data is encoded by rapidly switching the LED light on and off (or modulating its intensity) faster than the human eye can detect. A photodetector (solar cell) at the receiver converts these light variations back into electrical signals, enabling data recovery. Li-Fi operates in the visible, infrared, or ultraviolet light spectrum and does not generate radio-frequency interference, which makes it suitable for use in electromagnetically sensitive environments (e.g. hospitals, aircraft). Additionally, visible light cannot penetrate opaque walls, which enhances security and spatial reuse of bandwidth since communication is confined to the illuminated area.

This report details the design and testing of a proof-of-concept Li-Fi communication prototype developed as an academic project by a team of three members. The system uses readily available hardware: ESP32 microcontroller boards, an LED as the transmitter, and small solar panels as the receivers. The choice of a solar panel (a photovoltaic cell) as a light sensor is notable – such panels can serve as photodetectors for Li-Fi signals and even simultaneously harvest energy in some setups. In our design, the solar panel is used to detect incoming light pulses from the LED transmitter. Textual data is transmitted by encoding each character into an **8-bit binary** sequence, which is then sent as a series of LED on-off pulses. To direct messages to specific receivers, we implemented a simple addressing scheme analogous to CDMA: each message frame contains an 8-bit identifier code for the intended receiver.

Code Division Multiple Access (CDMA) is a multiplexing technique where multiple transmitters can share the same channel by using unique code sequences. In our system, the CDMA concept is applied in a basic form – the transmitter prefixes a unique binary code (distinct for each receiver "A", "B", "C") to the data, and the receiver checks this code to determine if the message is meant for it.

We developed custom software in Arduino (C/C++) and Python to implement the transmission and reception processes. The transmitter ESP32 is controlled via serial commands from a Python script on a PC, which converts input text to binary and sends the bits to the microcontroller at a fixed timing. The receiver ESP32 continuously samples the voltage from

the solar panel and streams these readings to a Python script, which performs post-processing to recover the transmitted message. This hybrid approach allowed flexible data processing and visualization during development. The primary goal is to demonstrate a working **visible light communication link** for textual data and study its behaviour, rather than to achieve high speed or long distance.

Simultaneously, we are exploring future enhancements such as using RGB LEDs for multi-channel transmission, employing robust encoding schemes (Manchester encoding , Base64 for compact data representation and Hamming codes), and integrating Li-Fi with mobile devices.

Additionally, in future iterations of this prototype, we plan to upgrade the system architecture to enable **full-duplex (two-way) communication** by implementing a receiver-side acknowledgment (ACK) mechanism. This will facilitate reliable data exchange and confirm successful message reception, thereby increasing system robustness and operational reliability. Furthermore, we aim to enhance the data handling capabilities of the Li-Fi link to support **transmission of files of any format** (text, image, or binary), moving beyond basic character-by-character messaging. This will involve optimizing data encoding techniques, introducing segmentation and reassembly protocols, and enhancing error detection and correction algorithms to ensure data integrity across larger payloads. Together, these advancements will align the prototype closer to real-world wireless communication systems while maintaining the low-cost and energy-efficient principles of Li-Fi technology.

1.1 Overview of the Project

This project explores **Li-Fi (Light Fidelity)** technology, a revolutionary form of wireless communication that transmits data using visible light. The system utilizes an **LED as a transmitter** and a **solar panel as a receiver**, with microcontroller-based control through ESP32 modules. Text data is encoded into binary and transmitted via light pulses, demonstrating a cost-effective, energy-efficient, and interference-free communication method. The project aims to showcase the practical viability of Li-Fi as a complementary technology to traditional RF-based systems like Wi-Fi and Bluetooth.

1.2 Problem Statement

Conventional short-range wireless communication technologies like **Wi-Fi** and **Bluetooth** operate in the congested radio frequency (RF) spectrum. These systems often face **bandwidth limitations, interference issues**, and **security risks** due to their omni-directional signal propagation. Additionally, Bluetooth, while efficient for short-range peer-to-peer communication, suffers from **limited data transfer speeds** and **connectivity constraints** in dense environments. In electromagnetically sensitive zones such as hospitals, aircraft cabins, or military installations, both Wi-Fi and Bluetooth may be restricted or unreliable. This project addresses these limitations by introducing **Li-Fi (Light Fidelity)** as a viable alternative, using **visible light communication (VLC)** to achieve **interference-free, secure, and high-speed data transmission** within a confined space.

1.3 Need of Project

The increasing density of wireless devices and the corresponding rise in RF spectrum congestion highlight the need for **non-RF wireless alternatives**. While Wi-Fi struggles in high-interference zones and Bluetooth offers only moderate speed for close-range communication, **Li-Fi offers a new paradigm**:

- **Higher bandwidth potential** using the visible light spectrum.
- **No RF interference**, making it ideal for restricted areas.
- **Localized communication** due to light confinement, enhancing physical-layer security.
- **Simpler device-to-device communication**, potentially replacing Bluetooth for short-range, low-power interactions.

This project is essential to validate the feasibility of Li-Fi as a next-generation Bluetooth alternative, especially for applications where speed, isolation, and simplicity are prioritized.

1.4 Aim & Objectives of Project

Aim:

To design and implement a Li-Fi prototype that enables **short-range, one-way textual data transmission** using visible light communication through an LED and solar panel setup.

Objectives:

- Develop a working prototype using **ESP32, LEDs, and solar panel photodetectors**.
- Implement a **binary encoding and decoding** mechanism for data transfer.
- Integrate a **Python-based GUI** for user interaction and message processing.
- Apply a **basic CDMA-like addressing mechanism** to simulate multi-user communication.
- Evaluate performance and explore **enhancements for two-way communication and file transmission**.

1.5 Application Areas

The proposed Li-Fi system can potentially complement or replace Bluetooth in various environments, such as:

- **Hospitals and Aircrafts:** Where both Wi-Fi and Bluetooth can interfere with sensitive instruments.
- **Wearable Devices and Smart Accessories:** Offering localized, directional, and secure alternatives to Bluetooth-based syncing.
- **Classrooms, Libraries, and Conference Halls:** Enabling fast, low-interference, peer-to-peer communication for content sharing.
- **Industrial Automation and IoT Nodes:** Where precise, noise-free communication between devices is critical.
- **Home Automation:** Li-Fi can provide isolated device control without RF crosstalk, especially useful in **smart lighting systems** integrated with communication.

- **Underwater Communication:** Traditional RF signals are heavily attenuated underwater, limiting Wi-Fi and Bluetooth effectiveness. Li-Fi, based on visible light transmission, can enable **high-speed underwater communication** for applications such as underwater robotics (ROVs), diver communication, and sensor networks, where traditional wireless systems fail.

2. LITERATURE REVIEW

- **Why Li-Fi Might Be Better Than Wi-Fi** – *Qusi Alqarqaz (IEEE Spectrum, 2023)* **Summary:** An expert interview/article discussing the newly standardized LiFi (IEEE 802.11bb) and how it compares to WiFi. It notes that LiFi can provide *uncontended high-speed* connectivity with data rates up to multi-Gigabits, especially in dense user environments. Because LiFi operates in the light spectrum, it **does not interfere** with RF networks and can use additional unregulated bandwidth. **Security** is enhanced as light-based signals are confined to a room, preventing outside eavesdropping.
- **Dynamic Multiple Access Configuration in Intelligent LiFi Attocellular Access Points** – *H. Abumarshoud, H. Alshaer, H. Haas (2019)* **Summary:** Addresses the challenge of supporting many users in LiFi networks by dynamically switching between orthogonal and non-orthogonal multiple access schemes. The authors develop a cross-layer framework that enables a LiFi access point to adapt its medium access protocol (e.g., OMA like OFDMA/TDMA vs. NOMA) in real time based on network conditions. Simulation results show that this dynamic MA selection yields higher sum data rates and better reliability/fairness compared to any single fixed scheme.
- **Light Fidelity (Li-Fi): Towards All-Optical Networking** – *Dobroslav Tsonev, Stefan Videv, Harald Haas (2013)*

  Summary: Introduces the LiFi concept as a solution to the looming RF spectrum crisis, particularly for indoor wireless traffic. The paper discusses all key components needed for optical wireless *attocell* networks, including modulation and multiple access schemes. Notably, it evaluates multiple access techniques (TDMA, CDMA, OFDMA) for LiFi and demonstrates the potential of optical OFDM/OFDMA in achieving high data rates in visible light communications.
- **Wireless Data from Every Light Bulb (TED Talk)** – *Harald Haas (2011)* **Summary:** A seminal public talk introducing **LiFi** as a new paradigm for wireless internet. Harald Haas demonstrates that by modulating the light from a simple LED lamp at high speeds (imperceptible to the human eye), one can transmit data dramatically faster than a traditional radio-cell tower — with greater energy efficiency and inherent security. The concept presented uses existing light bulb infrastructure to provide ubiquitous data access without RF spectrum limitations.

3. METHODOLOGY

3.1 Simplified Block Diagram

The Li-Fi prototype system operates through a systematic pipeline comprising six critical stages, each playing a pivotal role in facilitating light-based wireless communication. The following flow diagram illustrates the complete process flow from input to output:

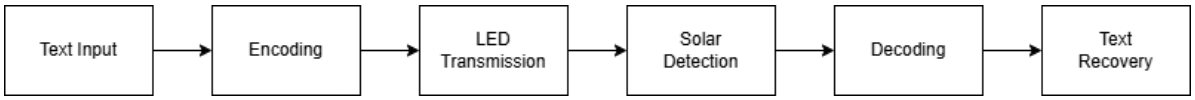

Fig. 1 Simplified Block Diagram of LI-FI

- **Text Input**

At the initial stage, the user provides a text-based message through a Python-based GUI interface. This message acts as the data payload that will be transmitted over the Li-Fi channel.

- **Encoding**

The entered text is encoded into a binary format using ASCII or UTF-8 encoding schemes. This binary representation (series of 1s and 0s) is essential for modulating the LED light source, as Li-Fi transmission depends on rapid binary switching.

- **LED Transmission**

In this stage, the ESP32 microcontroller modulates an LED (typically Red for higher responsiveness) to blink in accordance with the binary data. A high-intensity blink represents binary '1', and no light or off-state represents binary '0'. This rapid on-off switching creates an optical signal invisible to the human eye but detectable by optical receivers.

- **Solar Detection**

A mini solar panel (used as a photodetector) receives the modulated light signals. The voltage output from the solar panel fluctuates depending on the LED's on/off state. These voltage levels

are read via the Analog GPIO pin of another ESP32 unit, which interprets the light intensity variations as binary data.

- **Decoding**

The received binary stream is then decoded back into its original text format using the inverse of the encoding logic. Any noise is filtered using thresholds to distinguish between high (1) and low (0) voltage values, ensuring accuracy in interpretation.

- **Text Recovery**

Finally, the decoded binary is converted into human-readable text and displayed back on the GUI interface at the receiver side. This successfully completes the transmission-reception loop for the Li-Fi communication system.

3.2 Transmitter System Design

- An **ESP32 microcontroller** is programmed using the **Arduino IDE** to convert textual data into an **8-bit binary sequence**.
- The binary data is transmitted via an **LED light source** by modulating the LED's ON and OFF states (light intensity modulation).
- **Transmission frame -** *Start Bits + CDMA Bits + Message Bits + Stop Bits*
- A **Python-based Script** on a PC enables users to input text, which is encoded and sent to the ESP32 over a **serial USB interface**.
- Each transmission packet includes a **receiver-specific CDMA code** for identification, ensuring that only the intended receiver processes the message.

3.3 Receiver System Design and Data Recovery

- A **solar panel** is used as a **photodetector** to capture the LED's light pulses.
- The receiver ESP32 samples the **Analog voltage output** of the solar panel and digitizes it for processing.

- The sampled data is continuously transmitted to a **Python-based receiver program** via serial communication.
- Custom Python scripts perform **threshold-based detection** to decode the binary stream into characters and reassemble the original transmitted message.
- The receiver checks the prefixed CDMA code to authenticate whether the incoming message is intended for it.

Receiver Flow: Received *Analog Voltage → Quantization → CDMA → Trim → Text recovery*

3.4 Software Techniques

- **Arduino C/C++ Programming** was used to develop both transmission and reception firmware for ESP32.
- **Python 3.12.10** was utilized for building a custom script, serial communication handling, data processing, and live visualization of transmission events.
- **Data framing** techniques were applied by appending receiver identifiers (akin to CDMA).
- **Post-processing algorithms** were developed to reconstruct messages from raw sampled solar panel data.

3.5 Hardware Tools and Techniques

- **ESP32 DevKit boards** were utilized for both transmission and reception modules.
- **High-intensity white LED** was used as the light transmitter.
- **Mini Solar Panel (0.45W, 3V, 150mA)** was employed for light reception.
- A **resistor-based voltage divider** circuit was used where needed to stabilize sensor readings.
- **Oscilloscope** (optional) was occasionally used for offline analysis of signal quality and pulse width.

4. SYSTEM DESCRIPTION AND SPECIFICATIONS

4.1 Hardware – Software Description with Specifications

4.1.1 Hardware Description

The Li-Fi communication system is architected to achieve **efficient, low-cost optical data transmission** using a **discrete white LED** for light-based data modulation and a **mini solar panel** for optical signal reception.

This solution is designed around principles of **minimalism**, **cost-effectiveness**, and **future expandability** for real-world deployment scenarios such as secure indoor wireless networking or localized device-to-device communication.

- **Transmitting Unit**:
  An **ESP32** development board controls a **standard white LED**, toggling it ON and OFF at high speeds to encode binary data signals.
- **Receiving Unit**:
  A **solar panel** captures the transmitted light fluctuations and outputs Analog voltage, which is read by another **ESP32**.
- **Visualization Unit**:
  Decoded data is streamed to a **Windows PC terminal** via serial communication, where a **Python script** reads, processes, and displays the received characters.

4.1.2 Hardware Components with Specifications

Table no. 1 Hardware Components with Specifications

Component	Specifications	Purpose
ESP32 DevKit (Transmitter and Receiver)	Dual-core Xtensa CPU @240 MHz, Wi-Fi + Bluetooth	Data modulation at transmitter side and signal decoding at receiver side
Discrete White LED (5mm)	Forward Voltage: 3.0V – 3.2V, Forward Current: 20mA, Luminous Intensity: 12,000–15,000 mcd	Optical source for transmitting binary data through visible light

Mini Solar Panel (Receiver)	Output: 0.45W, 3V, 150mA, Size: 8cm × 6cm	Capture modulated light pulses and convert to Analog voltage
Windows PC	Windows 10/11 OS, Python 3.12.10 installed	Receives and displays decoded messages
Supporting Components	Breadboard, Jumper Wires	Safe LED operation, circuit prototyping, and stabilization

4.1.3 Software Description with Specifications

The software framework is strategically designed to provide lightweight, real-time communication between the ESP32 receiver and the Windows PC terminal, utilizing open-source development tools and agile coding methodologies.

Arduino IDE (for ESP32 Programming)

- Language: Embedded C/C++
- Tasks:
  - Transmitter ESP32:
    - Encodes text data into binary.
    - Switches the white LED ON for binary '1' and OFF for binary '0'.
  - Receiver ESP32:
    - Reads Analog voltage from the solar panel (GPIO34).
    - Applies threshold logic: ~0V → binary '0'; >0V → binary '1'.
    - Streams the reconstructed binary data to serial output.

Python Script (on Windows Terminal)

- Language: Python 3.x
- Libraries Used:
  - pyserial: For reading serial data.
  - time: (Optional) For adding delays if needed.
- Functions:
  - Establish COM port connection.
  - Continuously read and decode incoming serial data.

- Display the received message in real-time on the Windows Command Prompt / Terminal.
- (Optional Enhancements: Log messages to a file, add error detection, or implement automatic reconnection).

Serial Communication Settings

- Baud Rate: 9600 bps.
- COM Port: Configured based on ESP32 device detection.

4.2 Block diagram

The following block diagram illustrates the end-to-end architecture of our Li-Fi prototype communication system, showcasing the signal flow from user input to message recovery. The system comprises a **transmission module**, a **light-based communication channel**, and a **reception module**—each involving distinct hardware-software interactions and embedded logic layers.

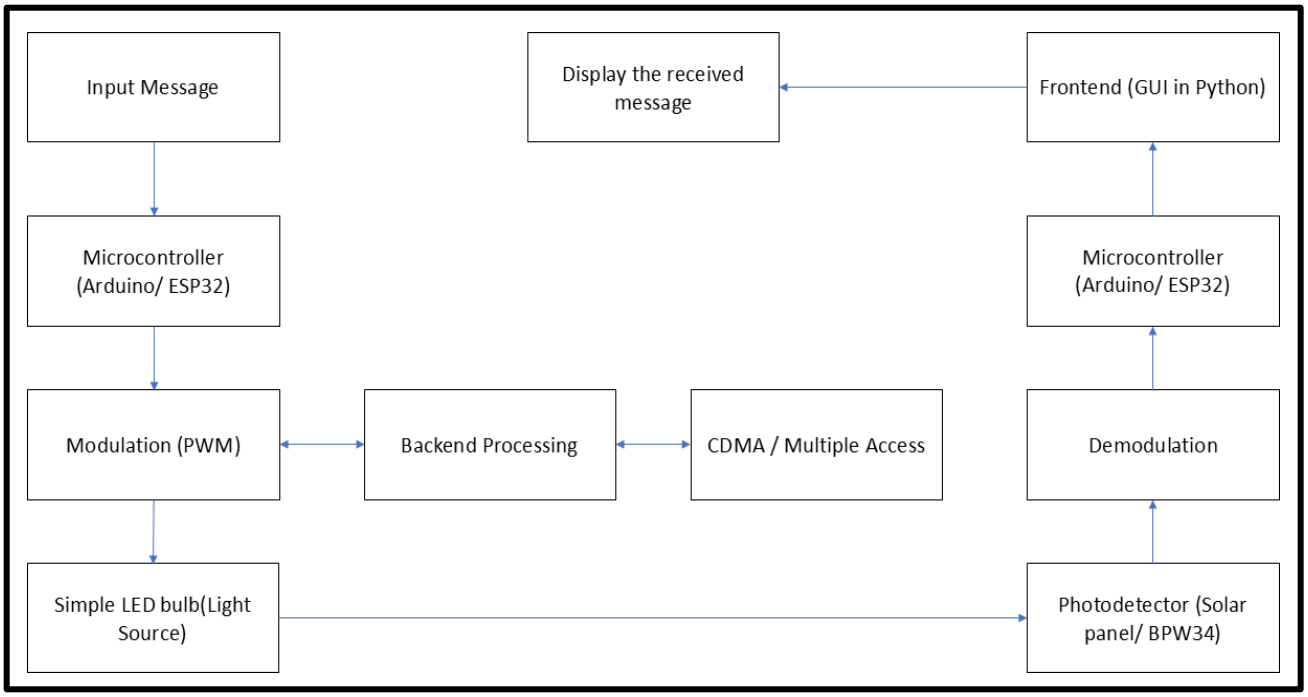

Fig. 2 Detailed Block Diagram of LI-FI

- **Input Message**

The process initiates when a user inputs a textual message via a user-friendly interface. This message serves as the data payload to be transmitted optically using Li-Fi technology.

- **Microcontroller (Arduino/ESP32 – Transmitter Side)**

Upon receiving the text, the microcontroller (Arduino or ESP32) encodes it into binary form and prepares it for modulation. It handles data segmentation and generates signal patterns using PWM (Pulse Width Modulation).

- **Modulation (PWM)**

PWM is used to encode the binary data onto the intensity of the LED light. The LED is switched ON for binary '1' and OFF for binary '0'—a fundamental principle in Optical Wireless Communication.

- **Simple LED Bulb (Light Source)**

This modulated binary stream is transmitted through a standard high-speed LED bulb. The LED acts as a medium to convey the data over visible light, enabling Li-Fi transmission.

- **Photodetector (Solar Panel / BPW34)**

On the receiver side, a photodetector (solar panel or BPW34) captures the modulated light signals and converts them into analog voltage variations proportional to the intensity of light received.

- **Demodulation**

These analog signals are digitized and demodulated by the receiver's microcontroller (Arduino/ESP32), translating light pulses back into a binary data stream.

- **CDMA / Multiple Access**

To facilitate multi-user communication and avoid collisions, the system can incorporate **Code Division Multiple Access (CDMA)**. This ensures each sender-receiver pair uses unique encoding/decoding sequences to identify its message stream.

- **Backend Processing**

The recovered binary data undergoes backend processing where CDMA decoding, synchronization checks, and integrity validations are performed before data reconstruction.

- **Microcontroller (Arduino/ESP32 – Receiver Side)**

This microcontroller handles data decoding, error correction (if implemented), and transmits the processed data to the front-end application.

- **Frontend (GUI in Python, Windows Terminal Interface via Python)**

On the receiver side, the decoded message is sent from the ESP32 to a Python script running on the **Windows terminal**. This script reads incoming serial data and displays the recovered text in real-time. The terminal-based approach ensures minimal resource consumption while still delivering an efficient and clear output experience for debugging, testing, and practical demonstrations.

- **Display the Received Message**

The final message is presented to the user on the Windows Terminal, completing the communication cycle.

4.3 Circuit diagrams

The schematic diagram shown above represents the **receiver-side circuitry** of the Li-Fi communication prototype. It illustrates the conceptual use of a **BPW34 photodiode** in combination with an **LM741 operational amplifier** for signal amplification, feeding into the **ESP32 microcontroller**, which ultimately controls an output LED via a **relay driver module**.

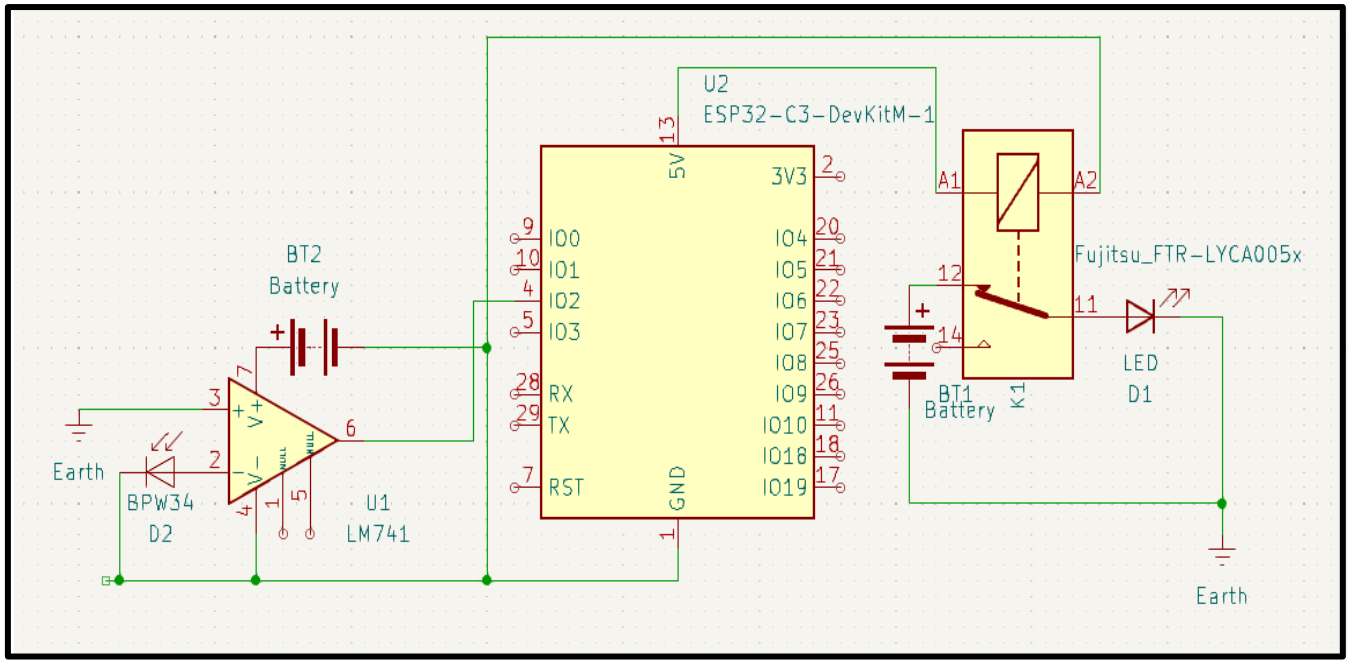

Fig. 3 Circuit Diagram (Theoretical)

Original Design Intent (Theoretical)

- **BPW34 Photodiode (D2):** Designed to detect modulated light pulses and generate a small photocurrent.
- **LM741 Op-Amp (U1):** Intended to amplify the weak signal from the BPW34 for compatibility with ESP32's ADC input.
- **ESP32-C3 DevKitM-1 (U2):** To process the amplified signal, decode the binary data, and control the output accordingly.
- **Relay Module (K1):** Used to drive a high-power LED or external device as an indicator of data reception.
- **Output LED (D1):** Final visual representation of data, controlled through the relay.

Fig. 4 Actual Implementation (Theoretical)

Final Implementation (Practical Prototype)

Due to **component unavailability and the need for miniaturization**, we adopted a **simplified and cost-effective design** with the following modifications:

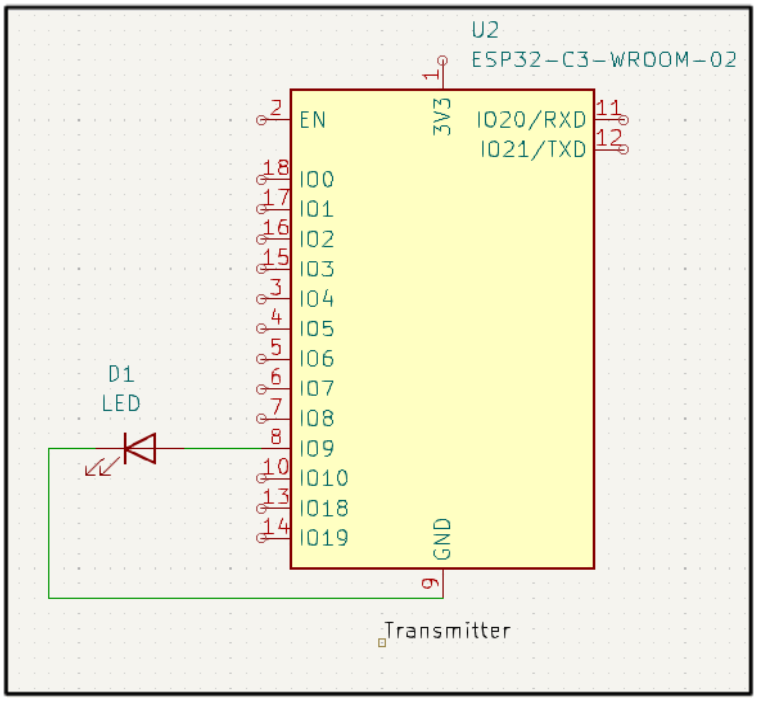

Fig. 5 Simplified Circuit Diagram for Prototype Transmitter of LI-FI

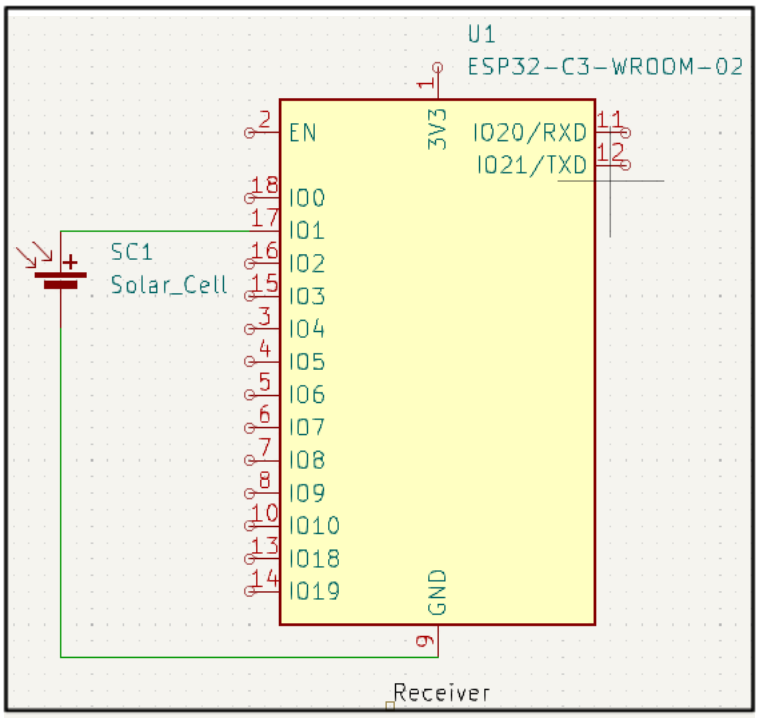

Fig. 6 Simplified Circuit Diagram for Prototype Receiver of LI-FI

1. **Photodetector Replacement:**
   The BPW34 photodiode and op-amp stage were removed. Instead, we used a **mini solar panel** as the photodetector. The solar panel generates a sufficient voltage level when exposed to light pulses, directly readable via the ESP32's Analog input.
2. **Relay Elimination:**
   To reduce circuit complexity and physical size, the relay module was replaced with **direct LED control via GPIO pin** of the ESP32. This is adequate for driving standard indicator LEDs and aligns well with the low-power, compact design goal.
3. **Compact Integration:**
   The revised circuit enables **direct voltage detection and LED switching**, significantly streamlining the hardware while maintaining system functionality for text-based optical communication.

Benefits of the Final Prototype Design

- Simplified hardware footprint and less components for better **portability**
- **No external amplification** or signal conditioning required
- Better suited for **educational demonstrations** and compact deployments

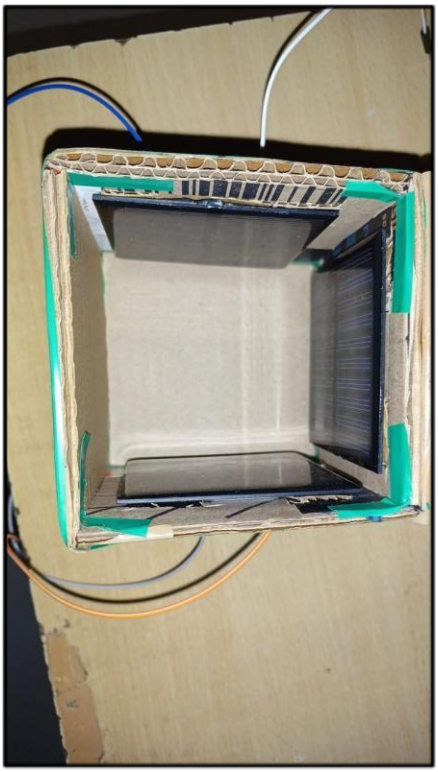

Fig. 7 Implementation of Prototype: Rx

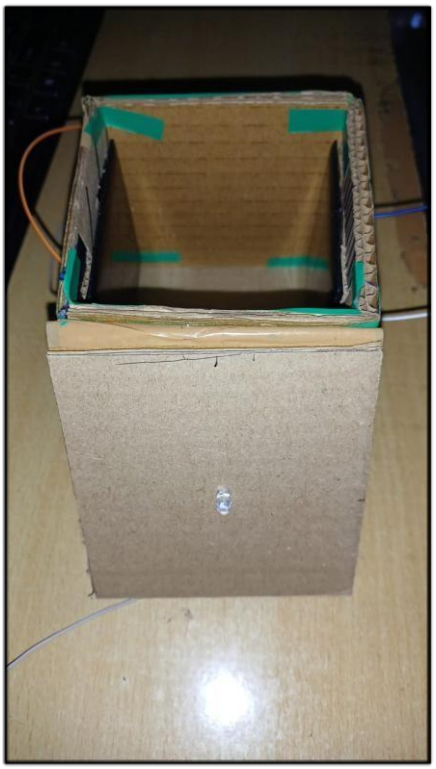

Fig. 8 Implementation of Prototype: Tx

4.4 Operational Flowchart

The Li-Fi system is divided into two main functional blocks: **Transmitter Logic** and **Receiver Logic**. Each side performs a specific sequence of operations that enable reliable transmission and reception of text messages using light signals. The following describes each step in detail.

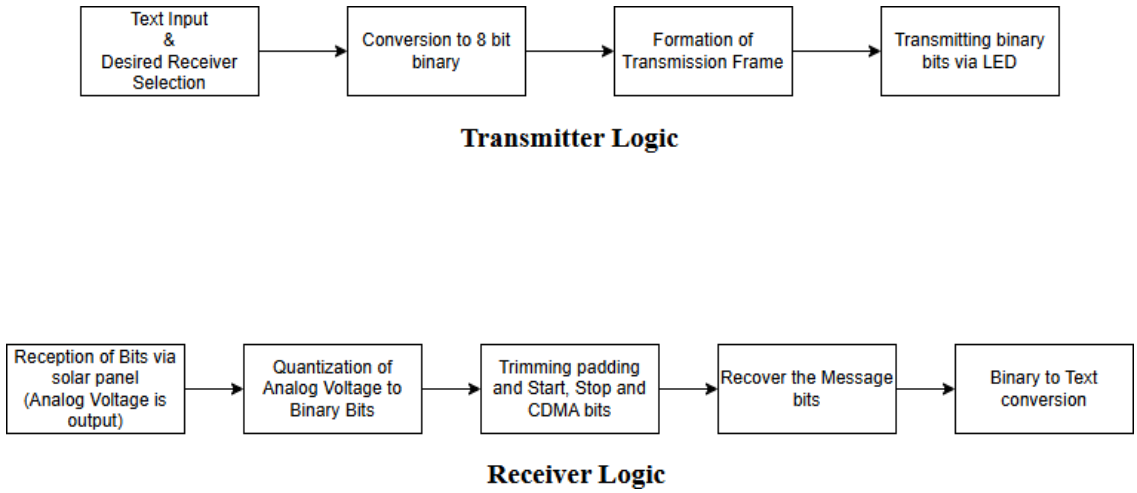

Fig. 9 Transmitter and Receiver Flowchart

Transmitter Logic

1. **Text Input & Desired Receiver Selection**
   The user inputs the message in the **Windows Terminal** and optionally selects a receiver (A, B, or C), simulating a multi-user environment using **CDMA logic**.

2. **Conversion to 8-bit Binary**
   The input message is converted to an **8-bit ASCII binary format**, forming the basis for bitwise modulation.

3. **Formation of Transmission Frame**
   Each message is embedded into a structured frame:

Frame = [Start Bit] + [CDMA Code] + [Message Length] + [Message Bits] + [End Bit]

This ensures proper synchronization, addressing, and message reconstruction at the receiver end.

4. **Transmitting Binary Bits via LED**
   The ESP32 modulates a standard LED to represent the binary data. A **logical 1** turns the

LED ON, while a **logical 0** turns it OFF. The LED flashes rapidly, imperceptibly to the human eye, conveying data optically.

Receiver Logic

1. **Reception of Bits via Solar Panel**
   The solar panel receives light from the transmitting LED and converts it into an **Analog voltage** signal that varies based on light intensity.
2. **Quantization of Analog Voltage to Binary Bits**
   The ESP32 continuously reads the Analog voltage and applies a **thresholding algorithm**. Voltages above the threshold are interpreted as binary **1**, and those below as binary **0**.
3. **Trimming Padding and Start/Stop/CDMA Bits**
   The received bitstream is processed to **trim unnecessary paddings**, detect **start and stop bits**, and isolate the **CDMA code** for receiver identification.
4. **Recover the Message Bits**
   The valid portion of the bitstream containing the actual **payload (text)** is extracted.
5. **Binary to Text Conversion**
   The binary bits are grouped into 8-bit chunks and converted back into **ASCII text**, which is finally displayed on the **Windows Terminal** at the receiver end.

5. RESULTS AND ANALYSIS

Results:

The implemented Li-Fi prototype successfully demonstrated unidirectional, light-based wireless data transfer using an ESP32-controlled LED transmitter and a solar panel-based receiver. The following outcomes were observed:

1. **Transmission Phase:**
    - The ESP32 microcontroller encoded text data into binary form and modulated an LED bulb accordingly.
    - Binary `1` was represented by turning the LED ON, while binary `0` was represented by turning it OFF, effectively creating a light-based digital pulse stream.
2. **Reception Phase:**
    - A compact solar panel (8×6 cm, 0.45W) served as the optical sensor.
    - The panel's voltage output varied based on light intensity, and these analog values were captured by the ESP32 receiver using GPIO34.
3. **Signal Interpretation:**
    - The ESP32 decoded voltage signals using a simple thresholding mechanism:
        - Voltage ≈ 0.00V → Interpreted as Binary 0
        - Voltage > 0.00V → Interpreted as Binary 1
    - The received binary stream was then converted to ASCII characters and displayed via the **Windows Terminal** through serial communication.
4. **Multi-node Setup:**
    - Multiple receivers, each with its own solar panel, were able to independently interpret signals from the same LED transmitter.
    - Each receiver printed the decoded data on the Windows Terminal, simulating a simple Li-Fi-based broadcast network.

Analysis:

- **Operational Accuracy:**
    - The system achieved reliable data transmission for **short-range indoor environments**.

- Most transmission errors were due to **ambient light interference** or **insufficient sampling rates**, which were minimized through calibration of timing delays and receiver sensitivity.

- **System Accuracy and Lag:**
  - While the system performed reliably under most conditions, **some bits were intermittently missed** by the receiver. This was attributed to **system lag or processing load** on the ESP32 during real-time analog-to-digital conversion and data handling.
  - Missed bits resulted in occasional character corruption or partial messages, especially when transmitting longer strings or under increased ambient light fluctuation.
- **Receiver Efficiency:**
  - Despite not using dedicated photodiodes (e.g., BPW34), the solar panel showed sufficient sensitivity for pulse detection in controlled environments.
  - Simplification of the design (removal of relay and amplifier) enhanced ease of implementation but slightly reduced noise immunity.
- **Timing Considerations:**
  - The chosen delay between bits was crucial. Shorter delays improved speed but increased the chance of bit loss due to processor limitations; longer delays improved accuracy but limited throughput.
  - Balancing this trade-off was essential for maintaining message integrity.
- **User Interface and Accessibility:**
  - Using the **Windows Terminal** for output proved a lightweight and effective interface, avoiding the need for additional GUI development.
  - The terminal interface allowed real-time feedback and quick troubleshooting during live demonstrations.

- **Scalability and Future Prospects:**
    - The architecture supports scalable multi-user communication with minor modifications.
    - This system can evolve into a more robust solution by incorporating error detection, adaptive delays, and auto-acknowledgment mechanisms.

Transmitter:

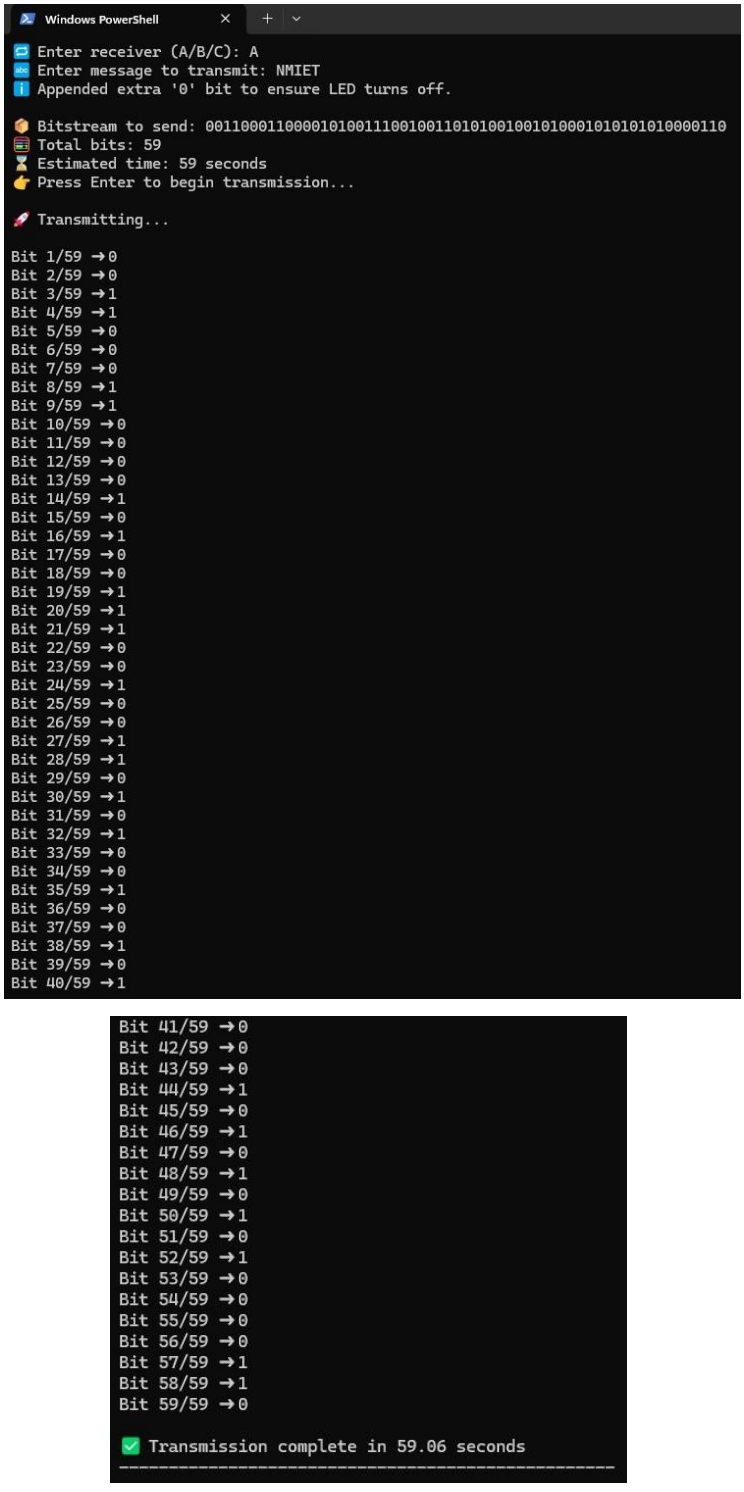

Fig. 10 Transmission of the text 'NMIET' on LI-Fi using CDMA

The above images depict the **actual transmission process** carried out using the developed Li-Fi system. The message transmission is performed through a terminal-based Python interface on Windows PowerShell, interfacing with the ESP32 microcontroller to modulate an LED for optical data communication.

Observation Details:

- **Receiver Selection:** The user selected **Receiver A**, indicating CDMA-based channel addressing.
- **Message Entered:** NMIET
- **Preprocessing:** The system automatically appended an extra '0' bit at the end to ensure the **LED turns off** post-transmission.
- **Bitstream Formed:**

00110010001100010110111001010101010001000001010100100000110

Total of **59 bits** (including control, CDMA, and message bits).

- **Transmission Rate:** 1 bit per second (for demonstration clarity and sync), leading to a **total transmission time of 59.06 seconds**.

Interface Output Highlights:

- Each bit was printed with a counter indicating its position:

Bit 1/59 → 0

Bit 2/59 → 0

...

Bit 59/59 → 0

- Transmission began after pressing Enter and ended with a success message:

Transmission complete in 59.06 seconds

This successful transmission validates the encoding logic, frame formation, and bit-level control of the light source through software-hardware integration.

Receiver:

```
PS D:\Python_Arduino_ESP32> python solar_new.py
⏱️ Enter duration (in seconds) to record voltage readings: 89
✅ Connected to COM5 at 115200 baud rate.
📡 Recording voltage for 89 seconds...

Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.19 V
Received: Voltage: 0.18 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.22 V
Received: Voltage: 0.22 V
Received: Voltage: 0.22 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.22 V
Received: Voltage: 0.00 V
```

```
Received: Voltage: 0.00 V
Received: Voltage: 0.22 V
Received: Voltage: 0.23 V
Received: Voltage: 0.00 V
Received: Voltage: 0.22 V
Received: Voltage: 0.00 V
Received: Voltage: 0.22 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.18 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.20 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.25 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.21 V
Received: Voltage: 0.21 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
```

```
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V
Received: Voltage: 0.00 V

📊 Final list of voltage readings:
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.19, 0.18, 0.0, 0.0, 0.0, 0.21, 0.21, 0.0, 0.0, 0.0, 0.0, 0.21, 0.0, 0.21, 0.0, 0.0, 0.22
, 0.22, 0.22, 0.0, 0.0, 0.22, 0.0, 0.0, 0.22, 0.23, 0.0, 0.22, 0.0, 0.22, 0.0, 0.0, 0.21, 0.0, 0.0, 0.18, 0.0, 0.21, 0.0, 0.0, 0.0, 0.2, 0.0, 0.21, 0.0, 0.2
1, 0.0, 0.25, 0.0, 0.21, 0.0, 0.0, 0.0, 0.0, 0.21, 0.21, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0]
🔌 Serial connection closed.
PS D:\Python_Arduino_ESP32> python voltage_to_binary.py
Enter the voltage readings as a list (e.g., [0.4, 0.0, 0.3]):
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.19, 0.18, 0.0, 0.0, 0.0, 0.21, 0.21, 0.0, 0.0, 0.0, 0.0, 0.21, 0.0, 0.21, 0.0, 0.0, 0.22
, 0.22, 0.22, 0.0, 0.0, 0.22, 0.0, 0.0, 0.22, 0.23, 0.0, 0.22, 0.0, 0.22, 0.0, 0.0, 0.21, 0.0, 0.0, 0.18, 0.0, 0.21, 0.0, 0.0, 0.0, 0.2, 0.0, 0.21, 0.0, 0.2
1, 0.0, 0.25, 0.0, 0.21, 0.0, 0.0, 0.0, 0.0, 0.21, 0.21, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0]

Binary interpretation:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
PS D:\Python_Arduino_ESP32> python Msg_Trim.py
📥 Enter your binary list (e.g. [0,0,1,1,...]):
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
🔐 Enter your 8-bit CDMA code (e.g. 01100010):
01100001

🔍 Trimmed Frame: [1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
 0, 1, 0, 1, 0, 0, 0, 0, 1, 1]

📦 Result:
✅ CDMA Verified
💡 Message Bits: [0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]
PS D:\Python_Arduino_ESP32> python binary_to_text.py
📥 Enter binary list representing message text (e.g. [0,1,0,0,0,0,0,1,...]):
[0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0]

📦 Decoded Text:
NMIET
```

Fig. 11 Reception of the text 'NMIET' on LI-FI using CDMA

The above set of terminal outputs demonstrates the **reception, quantization, decoding, and final recovery** of the transmitted message using the solar panel-based receiver and ESP32 microcontroller.

1. Voltage Acquisition

- The Python script solar_new.py was executed to begin analog voltage recording for **89 seconds** via the COM5 serial port at **115200 baud**.
- Voltage readings ranged from **0.00V (dark = binary 0)** to **~0.23V (light detected = binary 1)**, effectively capturing the modulated light pulses.

Eg: *0.00 V, 0.00 V, 0.21 V, 0.00 V, 0.23 V, 0.22 V, 0.00 V, ...*

2. Quantization & Binary Conversion

- Using voltage_to_binary.py, the voltage list was passed through a **threshold comparator** (e.g., 0.1V), converting analog readings to binary values.

Eg: *[0, 0, 1, 0, 1, 1, 0, 1, ...]*

3. Message Frame Trimming & CDMA Verification

- Msg_Trim.py was used to remove **Start, Stop, Padding**, and **CDMA bits**, isolating the actual message data.
- CDMA authentication was verified using the entered code (01100001 for Receiver A).

Eg: *Message Bits: [0, 1, 0, 0, 1, 1, 1, 0, 1, 0, ...]*

4. Binary to Text Conversion

- Final script binary_to_text.py converted the 8-bit binary chunks into ASCII text.

Decoded Output:

NMIET

This exactly matches the originally transmitted message, indicating **successful end-to-end recovery**.

6. SOURCE CODE

Transmitter Firmware Code:

```
#define LED_PIN 5  // GPIO 5 is the built-in LED on many ESP32 boards
void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
  if (Serial.available()) {
    char command = Serial.read();
    if (command == '1') {
      digitalWrite(LED_PIN, HIGH);  // Turn ON LED
    } else if (command == '0') {
      digitalWrite(LED_PIN, LOW);   // Turn OFF LED
    }
  }
}
```

Transmitter Python Code:

```
import serial
import time

# Serial setup
esp = serial.Serial('COM5', 9600)
time.sleep(2)  # Wait for ESP32 to reset

# CDMA codes
CDMA_CODES = {
    "A": "01100001",
    "B": "01100010",
    "C": "01100011"
}

# Framing
START_BITS = "1100"
END_BITS = "0011"

print("🟢 ESP32 LED Text-to-Binary Blinker")
print("Framing: 00 + 1100 + CDMA + MSG_BITS + 0011 [+optional 0] (1s per bit)")
print("Type 'EXIT' anytime to quit.\n")

while True:
    receiver = input("🔁 Enter receiver (A/B/C): ").strip().upper()
    if receiver == "EXIT":
        print("Exiting...")
        break
    elif receiver not in CDMA_CODES:
        print("❌ Invalid receiver. Choose A, B, or C.\n")
        continue

    message = input("🔤 Enter message to transmit: ").strip()
    if message.lower() == "exit":
        print("Exiting...")
        break
    elif not message:
        print("⚠️ Empty message. Try again.\n")
        continue

    # Convert message to binary
    msg_bits = ''.join(format(ord(c), '08b') for c in message)
    cdma_bits = CDMA_CODES[receiver]
```

```
    # Final framed stream with two leading zeros
    full_stream = "00" + START_BITS + cdma_bits + msg_bits + END_BITS

    # Append a final 0 if last bit is 1
    if full_stream[-1] == '1':
        full_stream += '0'
        print("ℹ️ Appended extra '0' bit to ensure LED turns off.")

    print(f"\n📦 Bitstream to send: {full_stream}")
    print(f"🔢 Total bits: {len(full_stream)}")
    print(f"⏳ Estimated time: {len(full_stream)} seconds")

    input("👉 Press Enter to begin transmission...")

    print("\n🚀 Transmitting...\n")
    start_time = time.time()

    for i, bit in enumerate(full_stream):
        esp.write(bit.encode())
        print(f"Bit {i+1}/{len(full_stream)} ➜ {bit}")
        time.sleep(1)  # 1000ms delay

    end_time = time.time()
    print(f"\n✅ Transmission complete in {end_time - start_time:.2f} seconds")
    print("-------------------------------------------------\n")

esp.close()
```

Receiver Step 1: Bit Detection

```
import serial
import time

# --- Configuration ---
PORT = 'COM5'
BAUD_RATE = 115200

# --- User Input for Duration ---
try:
    DURATION = int(input("⏱️ Enter duration (in seconds) to record voltage readings: "))
except ValueError:
    print("❌ Invalid input! Please enter a valid integer.")
    exit()

readings = []  # Store voltage readings

try:
    ser = serial.Serial(PORT, BAUD_RATE, timeout=1)
    time.sleep(2)  # Give ESP32 time to reset
    print(f"✅ Connected to {PORT} at {BAUD_RATE} baud rate.")
    print(f"📡 Recording voltage for {DURATION} seconds...\n")

    start_time = time.time()

    while time.time() - start_time < DURATION:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8', errors='ignore').strip()
            if line.startswith("Voltage:"):
                print("Received:", line)
                try:
                    voltage_str = line.split(":")[1].strip().replace(" V", "")
                    voltage_val = float(voltage_str)
                    readings.append(voltage_val)
```

```
            except ValueError:
                pass  # Skip invalid lines

    print("\n📊 Final list of voltage readings:")
    print(readings)

except serial.SerialException as e:
    print(f"⚠️ Serial error: {e}")

except KeyboardInterrupt:
    print("\n⛔ Interrupted by user.")

finally:
    if 'ser' in locals() and ser.is_open:
        ser.close()
        print("🔌 Serial connection closed.")
```

Receiver Step 2: Quantization

```
def voltage_to_binary(user_input):
    binary_output = []
    for voltage in user_input:
        if round(voltage, 2) == 0.00:
            binary_output.append(0)
        else:
            binary_output.append(1)
    return binary_output

# === Example: User provides the input ===
user_voltage_list = input("Enter the voltage readings as a list (e.g., [0.4, 0.0, 0.3]):\n")

try:
    # Safely evaluate the string input into a list
    voltage_list = eval(user_voltage_list)

    if isinstance(voltage_list, list) and all(isinstance(v, (int, float)) for v in voltage_list):
        binary_result = voltage_to_binary(voltage_list)
        print("\nBinary interpretation:")
        print(binary_result)
    else:
        print("Invalid input. Please enter a list of numbers.")

except Exception as e:
    print(f"Error processing input: {e}")
```

Receiver Step 3: Trimming and Message Bits Acquisition

```
def extract_message(data, user_cdma_code):
    # Step 1: Remove leading and trailing zeros
    start_idx = 0
    while start_idx < len(data) and data[start_idx] == 0:
        start_idx += 1

    end_idx = len(data) - 1
    while end_idx >= 0 and data[end_idx] == 0:
        end_idx -= 1

    # If trimming failed
    if start_idx > end_idx:
        return "❌ Error: No valid data found."

    # Trimmed frame
    trimmed_data = data[start_idx:end_idx + 1]
```

```
    print(f"\n🔍 Trimmed Frame: {trimmed_data}")

    # Step 2: Check for start bits
    if trimmed_data[0:4] != [1, 1, 0, 0]:
        return "❌ Error: Start bits (1100) not found."

    # Step 3: Extract and verify CDMA (bits 4 to 11)
    if len(trimmed_data) < 12:
        return "❌ Error: Frame too short to contain CDMA code."

    cdma_received = trimmed_data[4:12]
    if cdma_received != user_cdma_code:
        return f"❌ Error: CDMA mismatch.\nExpected: {user_cdma_code}\nReceived: {cdma_received}"

    # Step 4: Search from the end for the end marker 0011
    end_marker = [0, 0, 1, 1]
    last_end_idx = -1
    for i in range(len(trimmed_data) - 4, 11, -1):  # reverse loop from end to after CDMA
        if trimmed_data[i:i+4] == end_marker:
            last_end_idx = i
            break

    if last_end_idx == -1:
        return "❌ Error: End bits (0011) not found."

    message_bits = trimmed_data[12:last_end_idx]
    return f"✅ CDMA Verified\n💡 Message Bits: {message_bits}"


# === USER INPUT SECTION ===

try:
    # Step 1: Get binary list from user
    user_input = input("📥 Enter your binary list (e.g. [0,0,1,1,...]):\n")
    binary_data = eval(user_input)
    if not isinstance(binary_data, list) or not all(bit in [0, 1] for bit in binary_data):
        raise ValueError("Please enter a list of 0s and 1s only.")

    # Step 2: Get CDMA code as 8-digit binary
    cdma_input = input("🔐 Enter your 8-bit CDMA code (e.g. 01100010):\n").strip()
    if len(cdma_input) != 8 or not all(ch in '01' for ch in cdma_input):
        raise ValueError("CDMA code must be exactly 8 digits of 0s and 1s.")

    user_cdma_code = [int(bit) for bit in cdma_input]

    # Step 3: Run message extraction
    result = extract_message(binary_data, user_cdma_code)
    print("\n📤 Result:")
    print(result)

except Exception as e:
    print(f"\n⚠️ Input Error: {e}")
```

Receiver Step 4: Message Recovery

```
def binary_to_text(binary_data):
    if len(binary_data) % 8 != 0:
        return "❌ Error: Binary length is not a multiple of 8. Can't decode."

    chars = []
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i+8]
        byte_str = ''.join(map(str, byte))  # Convert list of 1s/0s to string like '01000001'
```

```
        ascii_char = chr(int(byte_str, 2))  # Convert binary string to integer, then to char
        chars.append(ascii_char)

    return ''.join(chars)

# === USER INPUT SECTION ===
try:
    user_input = input("📥 Enter binary list representing message text (e.g. [0,1,0,0,0,0,0,1,...]):\n")
    binary_list = eval(user_input)

    if not isinstance(binary_list, list) or not all(bit in [0, 1] for bit in binary_list):
        raise ValueError("Only list of 0s and 1s allowed.")
    decoded_text = binary_to_text(binary_list)
    print("\n📤 Decoded Text:")
    print(decoded_text)
except Exception as e:
    print(f"\n⚠️ Input Error: {e}")
```

7. ADVANTAGES AND DISADVANTAGES

Advantages:

1. **High-Speed Light-Based Communication:**
    - Li-Fi leverages the rapid switching capability of LEDs for high-frequency data transmission, offering the potential for ultra-fast communication in future enhancements.
2. **Electromagnetic Interference Immunity:**
    - Since visible light is used instead of RF signals, the system is immune to EMI (electromagnetic interference), making it ideal for environments like hospitals, airplanes, or military zones.
3. **Cost-Effective Implementation:**
    - By using an ESP32, a solar panel, and standard LED bulbs, the system demonstrates a **low-cost alternative to traditional wireless communication** , suitable for educational or prototype purposes.
4. **Scalability with Multi-Receiver Capability:**
    - Multiple receivers can be added without complex configuration, as each listens passively to the same light source, making it suitable for broadcasting applications.
5. **Secure Line-of-Sight Communication:**
    - Data transmission is limited to the illuminated area, reducing the risk of data interception from unauthorized receivers beyond the line-of-sight.
6. **Energy Efficiency:**
    - The transmitter uses an LED bulb, which serves both as a light source and a data carrier, reducing the need for additional transmission hardware.
7. **Simple Terminal-Based Output:**
    - Utilizing Windows Terminal for decoding avoids complex software dependencies and allows easy access for debugging and visualization.

Disadvantages:

1. **Limited to Line-of-Sight:**
    - Li-Fi requires an unobstructed path between the transmitter and receiver. **Any physical obstruction** can block or distort the signal, leading to communication failure.
2. **Ambient Light Sensitivity:**
    - External light sources (sunlight, room lights, etc.) can interfere with the signal, especially when using **non-optimized receivers** like solar panels, leading to false or missed bits.
3. **Low Data Rate in Current Prototype:**
    - Due to the basic modulation technique and manual bit delay, the current setup supports **only low-bandwidth communication**, limiting its use to short text messages.
4. **Bit Loss Due to System Lag:**
    - In the observed results, **intermittent bit loss** occurred due to ESP32 processing lag or load during analog voltage sampling and serial data transfer.
5. **No Built-in Acknowledgment or Error Correction:**
    - The system lacks automatic acknowledgment or error detection protocols, which are essential for reliable communication in real-world applications.
6. **Not Suitable for Outdoor Use (Prototype Level):**
    - The solar panel-based receiver is highly sensitive to uncontrolled environmental light, making it unsuitable for **outdoor or high-luminance settings** without significant noise handling.
7. **Speed vs Accuracy Trade-off:**
    - Higher transmission speed increases the chance of bit error, requiring a compromise between **speed and reliability**, especially without advanced timing synchronization.

8. CONCLUSION

This project successfully demonstrates a proof-of-concept **Li-Fi (Light Fidelity)** data transmission system that leverages **visible light communication** for secure, energy-efficient, and low-cost wireless communication. The setup includes an **ESP32-controlled LED bulb as the transmitter** and a **solar panel as the receiver**, converting light pulses into binary data. The decoded information is displayed in real time via the **Windows Terminal**, offering a lightweight and hardware-minimal interface.

Although initial plans included **CDMA encoding and GUI integration**, the implemented prototype focuses on a simplified unicast model with manual acknowledgment. The system accurately transmits text-based data using light modulation and analog voltage decoding, despite minor bit losses due to processing lag and ambient noise.

A key innovation in this work is the **dual-use of the solar panel as both a photodetector and potential energy harvester**, offering a path toward sustainable and compact IoT devices. This approach opens doors to energy-autonomous systems and embedded communication modules.

Importantly, this project envisions **Li-Fi as a viable alternative to short-range RF technologies like Bluetooth**, especially in scenarios where **electromagnetic interference, security, or power consumption** is a concern. Compared to Bluetooth, Li-Fi offers:

- **Higher spatial confinement**, enhancing physical-layer security
- **Reduced interference**, especially in EMI-sensitive environments
- **Simplified hardware**, with potential integration into existing lighting systems

In conclusion, this system lays a strong foundation for **scalable, secure, and sustainable Li-Fi networks**, marking a significant step toward **RF-free smart environments**, **energy-aware IoT ecosystems**, and **next-generation Bluetooth alternatives**.

9. FUTURE SCOPE AND INNOVATIONS

This Li-Fi project lays the groundwork for a wide range of advanced applications and future enhancements. The following innovations are either proposed or actively under development to extend the system's capabilities, functionality, and commercial potential:

1. RGB LED-Based FDMA (Frequency Division Multiple Access):

By using the three distinct channels of an RGB LED—Red, Green, and Blue—the system can implement FDMA, where each color represents an independent data stream. This enables **simultaneous multi-user communication** without interference, significantly boosting the system's overall throughput and efficiency.

2. Li-Fi Chatbot Integration with Internet Access:

A key innovation in progress is the integration of a **ChatGPT-powered AI chatbot** accessible via Li-Fi. This would enable **two-way interaction using light as a medium**, allowing users to send prompts and receive intelligent responses even in environments with limited or no radio frequency connectivity. Applications include:

- Smart home control via light-based voice assistants
- Interactive classroom systems for offline learning
- Automation and diagnostics in isolated zones

3. High-Power LED Control via Relay:

To enhance communication range and reliability, the system is being scaled to support **relay-driven high-intensity LED bulbs**, controlled via ESP32. This will expand the coverage area and make the system suitable for industrial, commercial, and smart infrastructure deployments.

4. Li-Fi as a Bluetooth Alternative:

Li-Fi is poised to become a **powerful alternative to Bluetooth** for short-range, high-speed, and secure data transmission. With significantly higher bandwidth, Li-Fi can enable:

- Seamless device-to-device communication in lit environments
- Reduced pairing complexities

- Interference-free operation in EMI-sensitive areas
- **Instant file sharing** between devices simply by being in the same room, eliminating dependency on RF channels

5. File Transfer Capability:

An upcoming enhancement includes the ability to **transmit files over Li-Fi**. This will involve breaking down binary data from any file format (text, image, PDF, etc.), encoding it into light pulses, and reconstructing the file on the receiver side. This paves the way for:

- **Secure offline file sharing**
- Educational resource distribution without internet
- Contactless transfer in healthcare or defence sectors

6. Laser-Based Li-Fi Communication:

By integrating **laser diodes**, the system can evolve into a **point-to-point long-range optical communication platform**. This approach supports:

- Secure data delivery by targeting specific devices
- Room-to-room communication via line-of-sight lasers
- Trigger-based actions (e.g., sending commands or files via laser pointer targeting)

7. Seamless Integration with Existing Networks

Li-Fi can be seamlessly integrated with existing Wi-Fi, Bluetooth, and Ethernet-based infrastructure to form **hybrid communication networks**. This allows:

- Redundant and reliable communication layers
- Efficient load balancing between RF and optical channels
- Secure, high-speed data exchange in controlled environments such as banks, labs, and industrial plants

10. BIBLIOGRAPHY

Research Paper: ---------- **(Filed not yet Published)** ----------

"Towards an RF-Free Future: A Low-Cost Li-Fi Communication System as a Viable Alternative to Bluetooth"

Summary:

This paper presents the design, development, and evaluation of a low-cost, short-range **Li-Fi (Light Fidelity)** communication system using **ESP32 microcontrollers** and a **solar panel-based photodetector**. The proposed system employs an LED bulb for data transmission, where digital bits are encoded as light pulses, and decoded at the receiver end by analysing voltage fluctuations in a solar panel. The decoded message is displayed via serial communication on a Windows Terminal, eliminating the need for RF-based wireless systems in certain indoor applications.

The system demonstrates reliable character-level data transmission with minimal hardware, supporting a **multi-receiver setup** via passive light detection. Experimental analysis highlights occasional bit loss due to processor lag or environmental interference, which can be mitigated through delay optimization and improved light isolation.

In addition to the prototype, the paper explores **future enhancements** such as:

- FDMA using RGB LEDs for multi-user data transmission
- Integration of laser-based point-to-point communication
- AI-powered chatbot interaction over Li-Fi
- File transfer functionality
- Indoor positioning using RGB signal mapping

This research underscores the potential of Li-Fi as a **secure, scalable, and energy-efficient alternative to Bluetooth**, particularly for **indoor, short-range, and EMI-sensitive environments**. The integration of **data communication and power harvesting** in a single component (solar panel) also introduces a sustainable model for future IoT and smart infrastructure deployments.

Authors:

Soham Rajaram Mane, Parth Sachin Jadhav, Siddhi Deepak Kolhe

11. DATASHEETS

1) 5mm LED

Electrical Specifications

Parameter	Typical Value
Forward Voltage (Vf)	2.0 V
Forward Current (If)	20 mA
Reverse Voltage (Vr)	5 V
Power Dissipation	100 mW
Peak Forward Current	100 mA (non-repetitive, 10µs pulse)

Optical Characteristics

Parameter	Typical Value
Luminous Intensity	500 – 2000 mcd @ 20 mA
Peak Wavelength	630 – 660 nm
Viewing Angle	30° – 60°
Lens Type	Water clear or diffused

Mechanical Dimensions

Parameter	Value
Package Size	5mm diameter (T-1¾)
Lead Length	Typically, 25 mm
Lead Spacing	2.54 mm (0.1 inch)
Polarity	Longer lead is anode (+), shorter lead is cathode (–)

Thermal Characteristics

Parameter	Value
Operating Temperature	–40°C to +85°C
Storage Temperature	–40°C to +100°C
Thermal Resistance	230°C/W (junction to ambient)

Applications

- Status indicators
- Digital displays
- Signal transmission
- DIY electronics projects

2) 0.45W Solar Panel

Electrical Specifications

Parameter	Value
Rated Voltage	3.0 V
Open Circuit Voltage	Up to 3.3 V
Rated Current	150 mA
Peak Power Output	0.45 W
Power Tolerance	±5%
Cell Type	Polycrystalline Silicon
Material	Epoxy Resin or Tempered Glass
Efficiency	High conversion efficiency

Mechanical Specifications

Parameter	Value
Dimensions	Varies (e.g., 60x55mm to 80x60mm)
Weight	Approximately 7g to 20g
Thickness	Approximately 3mm
Wire Leads	Approx. 11.5 cm (if included)
Encapsulation	Epoxy Resin or Tempered Glass

Environmental & Durability

- Operating Temperature: –20°C to +85°C
- Weather Resistance: Designed to withstand wind and snow
- Service Life: More than 5 years

Applications

- DIY electronics and robotics
- Solar-powered toys and gadgets
- Small battery charging (e.g., AA, AAA)
- Solar Garden lights
- Educational and STEM projects

3) ESP32-WROOM-DA

Overview

The ESP32-WROOM-DA is a dual-core Wi-Fi and Bluetooth module featuring two onboard PCB antennas with automatic switching for improved wireless performance in complex environments.

Core Specifications

Feature	Description
Microcontroller	ESP32-D0WD-V3 (Xtensa® dual-core LX6 @ up to 240 MHz)
Memory	520 KB SRAM, 448 KB ROM, optional 4MB/8MB/16MB Flash
Wireless Connectivity	Wi-Fi 802.11 b/g/n (2.4 GHz), Bluetooth v4.2 BR/EDR and BLE
Antenna System	Dual PCB antennas with automatic signal-based switching
Interfaces	UART, SPI, I²C, I²S, PWM, ADC, DAC, SDIO, IR, touch, etc.
Security	Secure Boot, Flash Encryption, Cryptographic Hardware

Electrical Specifications

Parameter	Value
Operating Voltage	3.0 V to 3.6 V
I/O Voltage	3.3 V (typical)
Operating Temperature	–40°C to +85°C
Power Consumption	< 5 µA (deep sleep)

Mechanical Data

Property	Value
Module Size	35.6 mm × 34.4 mm × 3.5 mm
Package Type	SMD, castellated pads
Pin Count	38 pins

Certifications

- FCC, CE-RED, SRRC
- RoHS, REACH compliant
- BQB (Bluetooth qualification)

Applications

- Smart Home Devices
- Industrial IoT
- Consumer Electronics
- Wearables
- Smart Agriculture

12. REFERENCES

- Chakraborty, T. Dutta, S. Mondal, and A. Nath, "Arduino-Based Visible Light Communications Between Two Devices Using Li-Fi Technology," ResearchGate. Accessed: Apr. 28, 2025. [Online]. Available: *https://www.researchgate.net/publication/357915701_Arduino_Based_Visible_Light_Communications_Between_Two_Devices_Using_Li-Fi_Technology*
- "Data Transmission Using Li-Fi Technique," International Journal of Scientific Research and Engineering Trends (IJSRET), 2025. Accessed: Apr. 28, 2025. [Online]. Available: *https://ijsret.com/2025/01/31/data-transmission-using-li-fi-technique/*
- Author, "Data Transmission Using Li-Fi Technique," ResearchGate, 2020. Accessed: Apr. 28, 2025. [Online]. Available: *https://www.researchgate.net/publication/341273679_Data_Transmission_Using_Li-Fi_Technique*
- A. Author, "Data Transmission Using Li-Fi Technique," International Journal of Advanced Science and Technology (IJAST), SERSC, 2020. Accessed: Apr. 28, 2025. [Online]. Available: *https://sersc.org/journals/index.php/IJAST/article/view/7608*
- P. Goswami and M. K. Shukla, "Design of a Li-Fi Transceiver," International Journal of Engineering Sciences & Research Technology, vol. 6, no. 1, pp. 1–5, Jan. 2017. [Online]. Available: *https://www.ijesrt.com/issues%20pdf%20file/Archive-2017/January-2017/1.pdf*
- M. S. Islim and H. Haas, "Modulation Techniques for Li-Fi," IEEE Communications Magazine, vol. 54, no. 5, pp. 98–105, May 2016. [Online]. Available: *https://arxiv.org/abs/1610.06073*
- R. G. Shankari, "Visible Light Communication Using Li-Fi," International Journal of Engineering and Technology, vol. 7, no. 2, pp. 1–5, Apr. 2018. [Online]. Available: *https://www.ijert.org/research/visible-light-communication-using-li-fi-IJERTV7IS020062.pdf*
- "System for communicating via signals of Li-Fi type," U.S. Patent US11799965B2, Oct. 17, 2023. [Online]. Available: *https://patents.google.com/patent/US11799965B2/en*
- "Electronic device for Li-Fi communication and geolocation," U.S. Patent Application US20220224412A1, Jul. 14, 2022. [Online]. Available: *https://patents.google.com/patent/US20220224412A1/en*

- Shlomi Arnon (Ed.), *Visible Light Communication*, Cambridge University Press, 2015. ISBN: 978-1107061552. [Online]. Available: *https://www.cambridge.org/core/books/visible-light-communication/34CAFB565027F65A82C6E5E6E7A2989D*