OptiCOMM: A Multi-Channel Light Communication System

Ujas Shah[1]

[1]Computer Science and Engineering, Indian Institute of Technology Gandhinagar, Gandhinagar-382355, India

***Abstract*—This paper presents the development of *OptiCOMM*, a light-based 4-channel binary communication system powered by an Arduino UNO R3. The system utilises four LEDs as transmitters, corresponding Light Dependent Resistors (LDRs) as receivers, and a clock LED to synchronise transmission and reception. Additionally, the system stores the last transmitted message in EEPROM memory, enabling automatic replay upon restart. This project demonstrates a low-cost, short-range optical communication system that illustrates key concepts such as parallel optical communication, channel-based data encoding, optoelectronic signal conversion, analog-to-digital conversion (ADC), thresholding, and clock synchronisation. Performance evaluation shows reliable message decoding under controlled conditions, emphasising the potential for further development into more robust optical communication systems.**

Keywords — Light-based, 4-channel, binary communication system, transmission-reception, EEPROM, short-range optical communication system.

I. Introduction

The concepts of ADC and binary conversions taught in lectures in ES 116 / Principles and Applications of Electrical Engineering course at IIT Gandhinagar have motivated the project. There arose a keen interest in how communication can happen in just the terms of 1s and 0s, ONs and OFFs. To quench this thirst of curiosity, this project was chosen as the course project to know how bit communication takes place, and work on how its speed can be affected and improved. Four pairs of LEDs and LDRs were connected with corresponding resistors to an Arduino UNO R3; LEDs to digital pins and the LDRs to analog pins. Upon input of a message, the Arduino Code would convert each character to its binary ASCII equivalent, which the LEDs would then transmit. A clock LED is used to mark the transmission and reception periods, to synchronise the transmission and reception process. Received by the LDRs as light intensities, it would be again encoded to digital from analog (Analog-to-digital conversion) using a threshold as a mark to differentiate between ON and OFF. Once converted to 8-bit binary ASCII code, it would again be transformed into its character equivalent. An output would be produced with the received message and the total time taken in the process in milliseconds. It demonstrates concepts like Binary communication and ADC along with clock synchronisation.

II. Methodology

First, we started with a one LED-LDR pair circuit simulation in Tinkercad Circuits to understand the working of a LDR along with an LED, how it gives an output, threshold, etc. Fig 1 shows the circuit used to first grasp the basic concepts of LDR working.

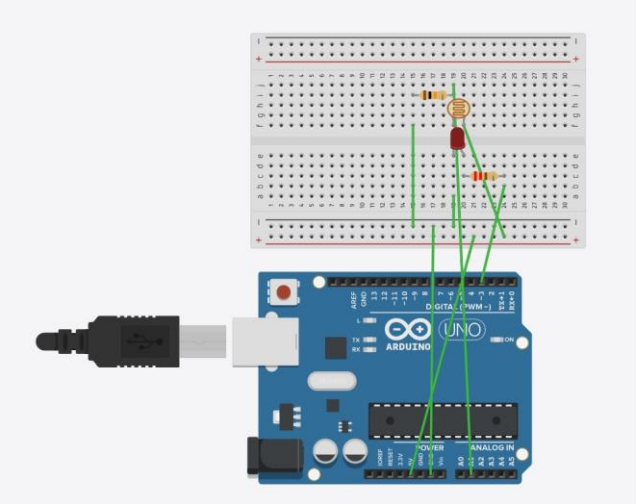

Fig 1. Basic one LED-LDR pair circuit

According to the standard use guidelines, a 220-ohm resistor was used with the LEDs and a 10-kohm resistor with the LDR for proper output values. Upon gaining confidence in the simple circuit, we made the 4-channel transmission circuit in Tinkercad, as shown in Fig. 2.

Once the simulation was successful, we started making the actual circuit physically. The following were used as components:

- Arduino UNO R3
- Male to Male jumper wires
- 4 white LEDs (for transmission)
- 4 LDRs (Photoresistors)
- 1 red LED (clock)
- USB cable to connect Arduino and Laptop
- Five 220-ohm resistors
- Four 10-kohm resistors
- Arduino IDE (to code)

After connecting the LEDs and LDRs accordingly, the very first thing to do was to test the circuit. So we ran single-channel transmission through all circuits individually, to test each part of the circuit. Once sure all connections were correct, we tested our code and circuit for 4-channel transmission.

What we have implemented is simple. Once a message is

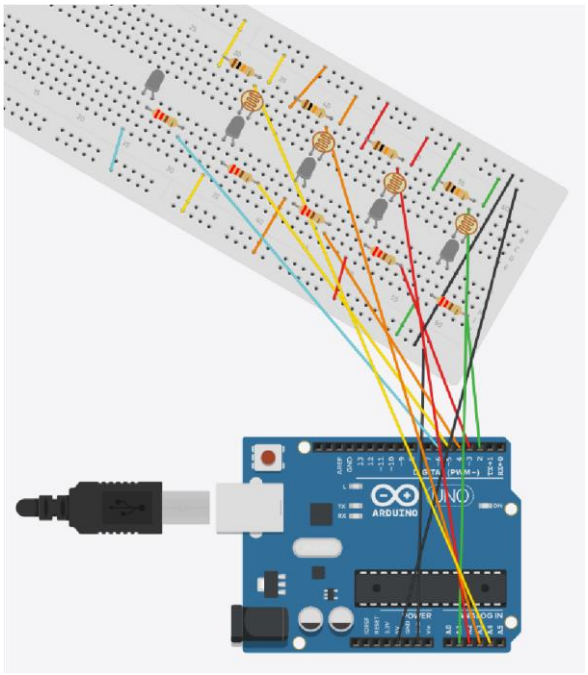

Fig 2. Final Project Circuit

entered, each character is taken at a time, and is converted to its 8-bit ASCII equivalent. Now, since we have four channels, the 8-bit ASCII code is broken into two parts, and 4 bits of each part are transmitted simultaneously through the four channels. The clock LED synchronises when the transmission happens, and when are the LDRs supposed to receive. Once the LDR reads the intensities, they classify them into 1 or 0 according to a set threshold. Then these bits are again collected back into groups of 8 and transformed to their character equivalent. Once all the characters have been transmitted and received, they are printed on the serial monitor as the output.

A severe problem here was the threshold. It tended to change or deviate in different lighting conditions, which resulted in wrong outputs under certain conditions. So we updated the code to first take three readings of ON and OFF for each LDR, average them and take their mean as the threshold for the transmission till the Arduino is restarted. This ensured that we always have the correct threshold values set, and the output does not go wrong.

We also used EEPROM to save the last input given in the Arduino, to have a non volatile memory, so we can revisit the last message sent whenever the reset button is pressed or the Arduino is restarted. All this hard work did yield exciting and encouraging results.

III. Results

The project fulfilled its objective. We were successfully able to take input messages from the user, set the threshold values for each channel experimentally, transmit the message through the four LED channels, receive and convert the analog light intensity values to digital values of 1 and 0 to represent the ASCII binary for transmitted character, output the transmitted message along with the time taken in the transmission and reception process. Also we used EEPROM to retain the last message sent, and repeat its transmission if the reset button on the Arduino is pressed.

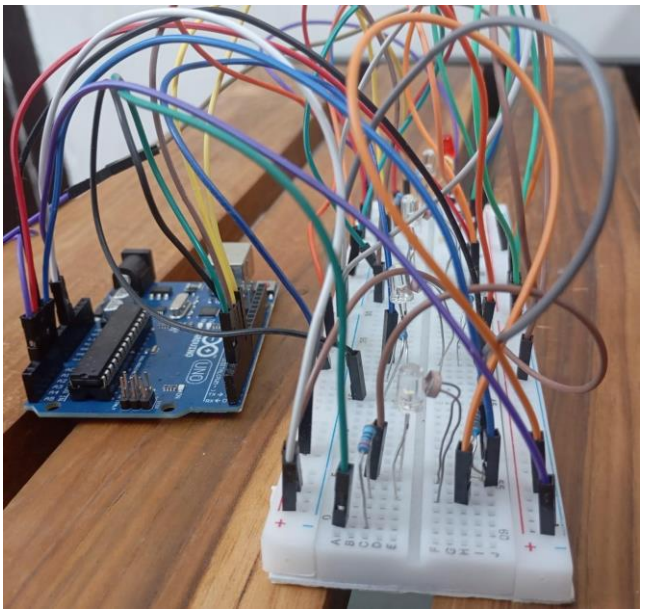

Fig 3. Our project

IV. Conclusion

This project successfully demonstrated the implementation of a simple yet effective 4-channel optical communication system using LEDs and LDRs, controlled via an Arduino UNO R3. By converting characters into binary ASCII format and transmitting the data through visible light, the system offered a practical understanding of binary communication, analog-to-digital conversion (ADC), and synchronization using a clock signal.

The project not only reinforced core concepts taught in the ES 116 course at IIT Gandhinagar, such as digital signal encoding and embedded systems interfacing, but also provided hands-on experience in circuit design, simulation, and troubleshooting. The integration of EEPROM memory allowed message retention across resets, while the introduction of a dynamic threshold calibration mechanism significantly improved the accuracy and reliability of data transmission under varying ambient light conditions.

Overall, this project served as an insightful exploration into the fundamentals of optical data transmission and showcased how relatively simple electronic components can be combined to build a functional communication system. It lays the foundation for further enhancements such as increased channel count, longer transmission distance, noise reduction, or even integrating photodiodes and IR LEDs for more robust and faster communication.