Table of Contents

1. Abstract

Optical Camera Communication (OCC) is a form of visible light communication where data is sent using LED lights and received by a camera. Instead of radio signals, light intensity is used to carry information. This project demonstrates a simple OCC system using a regular webcam as the receiver and an LED array as the transmitter. The LED blinks very fast in specific patterns that represent digital data, and the camera captures these patterns as a video. A Python program then processes the frames in real time to extract the data. The project also uses Manchester coding to make communication more reliable and avoid flickering. The purpose of this project is to show that even without expensive optical hardware, a working OCC system can be built using common electronic parts and free software tools.

2. Introduction

Every year, more and more devices connect to the internet through wireless signals. Most of these devices use radio frequency (RF) communication such as Wi-Fi and Bluetooth. However, RF frequencies are becoming crowded, and there are limits on how much data can be transmitted.

To solve this, engineers have been exploring Visible Light Communication (VLC) — a technology that uses LED light to transmit information. Optical Camera Communication (OCC) is one branch of VLC that uses LEDs as transmitters and cameras as receivers.

The idea is simple: a light source turns ON and OFF very quickly — faster than human eyes can detect. These light pulses can represent binary 1s and 0s. A camera captures the blinking pattern as a video, and computer software decodes it to recover the data.

Webcams capture images line by line (a property called the rolling shutter effect). This effect causes the LED flickering to appear as white and black strip patterns in the image, which are then decoded to get back the data.

This project demonstrates a low-cost implementation of OCC using a common Logitech C922 webcam and a custom-made LED transmitter panel. The aim is to help students, hobbyists, and beginners understand how OCC works and how they can build their own version using affordable components.

The application of this type of communication can be found in:

- Indoor secure data transfer (e.g., device-to-device communication).
- Localization and IoT applications (interference-free with RF systems).
- Vehicle-to-everything (V2X) and smart lighting systems using existing infrastructure.

3. Components

3.1 Arduino Mega 2560

The Arduino Mega 2560, the successor to the Arduino Mega, is a microcontroller board based on a ATmega2560 AVR microcontroller. It has 70 digital input/output pins (of which 14 can be used as PWM outputs and 16 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. The Arduino has an extensive support community, which makes it a very easy way to get started working with embedded electronics. [1]

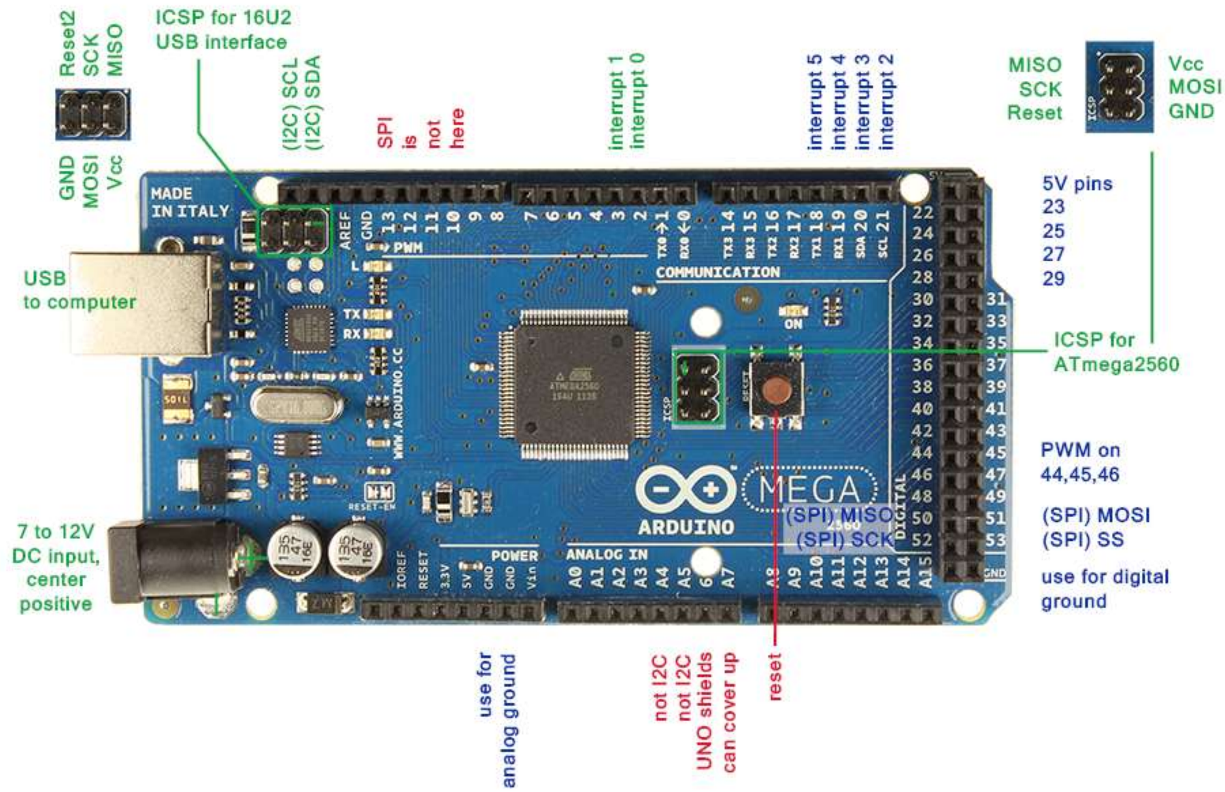


Figure 1: Arduino Mega 2560

3.2 LED Array

The LED array acts as the transmitter. In this project, 236 white LEDs were arranged on a perforated board (barro board) of about 6 cm × 32 cm. Each LED group receives 3 volts, and four LEDs are connected in series. Multiple such groups are connected in parallel to share the 12V power supply. A TIP122 transistor is used as a driver so that the Arduino does not need to supply large current directly. When the Arduino sends a digital HIGH signal, the transistor switches ON the LED array.

(a) Transmitter LED Array

(b) Shielding for uniform lighting

(c) Transmitter on power

Figure 2: LED Array

3.3 Webcam (Logitech C922)

The Logitech C922 Pro Full HD Webcam can create 720p at 60 fps and 1080p at 30 fps videos. It also comes with background replacement feature and autofocus technology. It can be connected directly to a computer through an USB interface and can be tuned using LogiTune software. The camera is discoverable as an external device and can be read from using python OpenCV. [2]

Figure 3: Webcam (Logitech C922)

3.4 DC Power Adapter

A 12V DC adapter is used to power the LED array. The Arduino itself is powered through a USB connection to the computer.

Figure 4: 12V DC Adapter

3.5 Bread Board

A modern solderless breadboard socket (invented by Ronald J Portugal for E&L Instruments, Derby CT) consists of a perforated block of plastic with numerous tin-plated phosphor bronze or nickel silver alloy spring clips under the perforations. The spacing between the clips (lead pitch) is typically 0.1 inches (2.54 mm). Integrated circuits (ICs) in dual in-line packages (DIPs) can be inserted to straddle the centerline of the block. Interconnecting wires and the leads of discrete components (such as capacitors, resistors, and inductors) can be inserted into the remaining free holes to complete the circuit. Typically, the spring clips are rated for 1 ampere at 5 volts and 0.333 amperes at 15 volts (5 watts). [3]

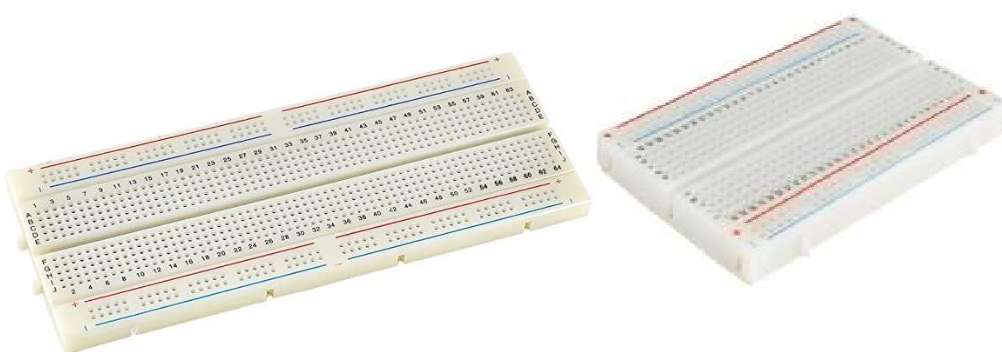


Figure 5: Bread Boards

3.6 Jumper Wires

A jumper wire is a conducting wire used to transfer electrical signals between two points in a circuit**.** The wires can either be used to modify circuits or to diagnose problems within a circuit. Jumper wires typically vary in color and size depending on what they are being used for. In breadboards, jump wires are used to establish connections between the central micro controller and other devices such as buttons and sensors. If possible, the jumper wire should always be placed on the component side of a circuit board during assembly. The wires should also be routed in an X-Y manner, avoiding any bends. Jump wires should never be raised more than 1/8 of an inch above the surface of the circuit board.

Figure 6: Jumper Wires

4. Block Diagram

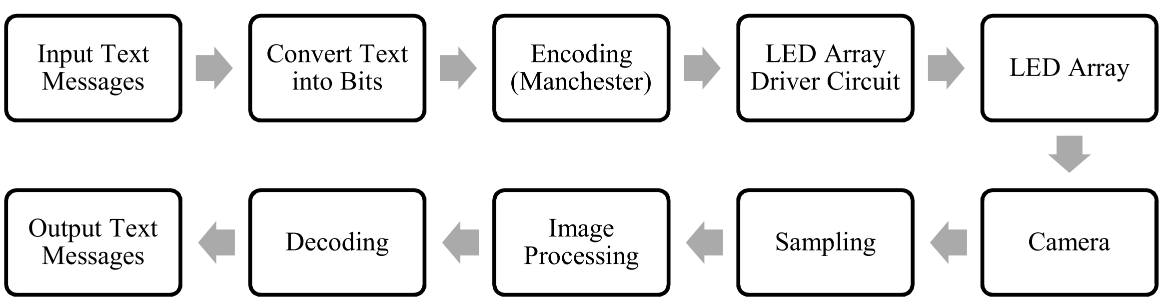
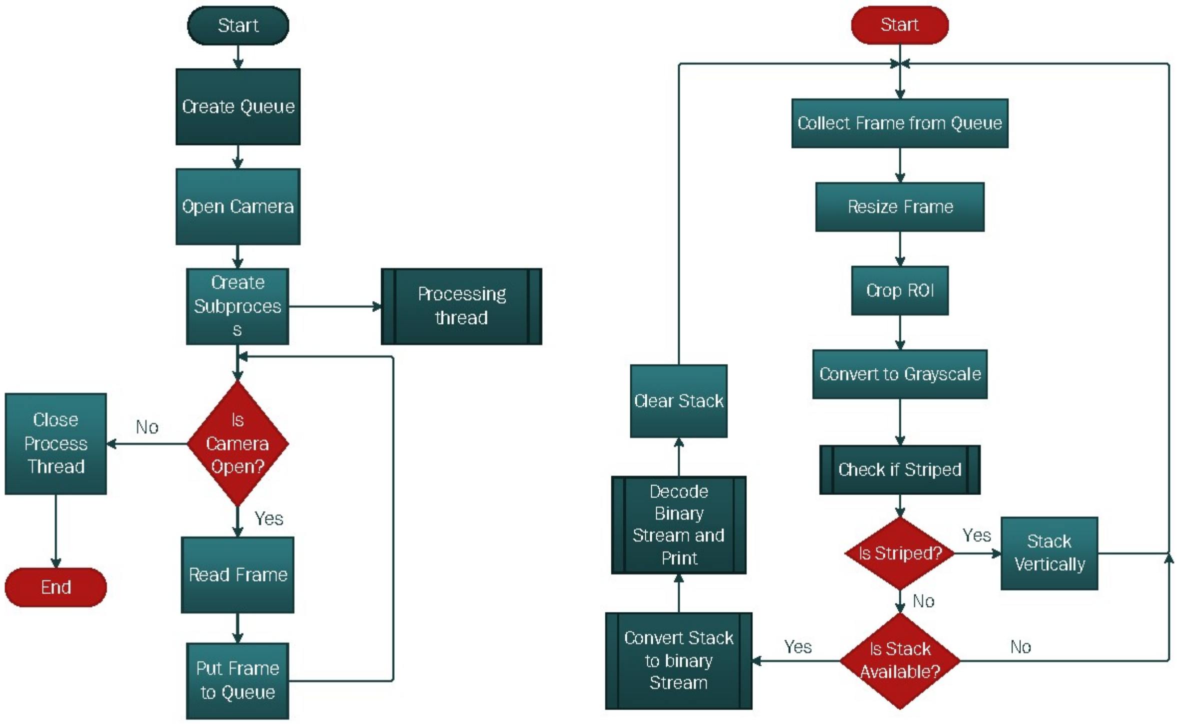


Figure 7: Block Diagram of OCC

5. Methodology



(a) Main Process (b) Converter Subprocess

Figure 8: Flow Chart of Decoder Program

The system works by blinking the LEDs at high speed to represent data. Each ON/OFF blink corresponds to binary values 1 and 0. These blinks happen much faster than the human eye can detect, so the light looks steady — but the camera can still capture the changes.

5.1 Data Encoding

The Arduino program takes a text message (like "HELLO"). Each character is converted into binary form. To make communication stable, the project uses Manchester coding. This means every bit (0 or 1) is represented by a transition. For example, "1" means the LED turns ON then OFF, and "0" means the LED turns OFF then ON. Therefore, a byte or 8 bits of data is converted into a 16 bits long code. For the sake of synchronization, each packet of 16 bits is preceded by a 0 bit and succeeded by a 0 bit. So, the data packet ultimately becomes 18 bits long. The good thing about Manchester coding is that only two consecutive bits can be of the same type. This can reduce error in transmission, with a tradeoff of an increase in data size.

5.2 Light Transmission

The encoded data drives the LED array using the transistor circuit. The LEDs flash rapidly to send the data pattern into the air as light signals. To get a smooth shine from the panel, a semitransparent plastic shielding is used to cover the front. The panel is placed at a short distance from the camera, and the background light is minimized as much as possible to increase the sharpness of the strip edges.

5.3 Video Capture

The camera is placed directly in front of the LED array, facing it from a few centimeters away. It records the LED blinking as a continuous video feed. Because the camera scans rows one by one (rolling shutter effect), the captured frames contain bright and dark strips representing the transmitted bits.

5.4 Frame Analysis

Each video frame is analyzed by the Python program. In the beginning, sample information is transmitted for calibration purposes. A calibration program analyzes the video feed to determine the width of the strips. From the output of this program, the calibration parameters are set in the decoder program. Events like change of distance, background lights, frame rate etc. can affect the strip width, and re-calibration may be required.

After calibration, the program starts collecting the frames from the webcam. Any frame that contains strips is first detected and put into a queue in the computer memory. The program has a background thread that collects these frames from the queue, cuts a small Region of Interest (ROI), converts it to grayscale, and stacks it on any previously processed frames. There must be at least one blank frame between each byte of data, because when the program detects a blank frame, it decodes any message that has been stacked up.

5.5 Data Decoding

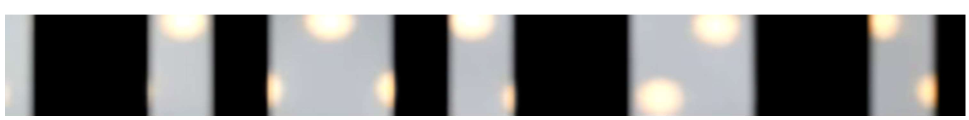
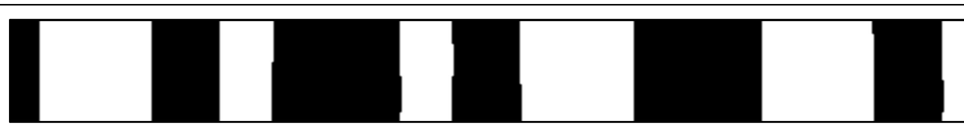
The first step of decoding is to measure the width of the strips within the stacked image. The calibration values contain the width of one black and one white strip. So dividing any width value with the corresponding strip size gives the number of bits within the strip. Thus, a nice stream of bits is obtained from the stacked image, from which the actual binary data is recovered using the same rule as before, “01” for 0 and “10” for 1. The leading and trailing synchronization 0 bits are ignored. The binary data is then converted into text and displayed in real time.

There is an issue known as shutter loss in webcams, where some portion of a strip width is lost during the transition from one frame to another. This problem puts an upper limit to frame rate, because with a higher frame rate, the strip width is smaller and thus shutter loss is more significant. It is addressed by the calibration program, which also gives the value of shutter loss in its output, which is added in case of an unusual strip width. Due to shutter loss and other effects, the higher the number of bits embedded within a single strip, the higher the uncertainty about the number.

This is why Manchester coding is useful in this case; the highest number of the same type of consecutive bits is 2. To reduce the errors further, each byte is sent multiple times.

6. Practical Implementation

Figure 9: Practical Implementation

<table>
<tr><td>ROI Cropped</td><td colspan="12"></td></tr>
<tr><td>Grayscale</td><td colspan="12"></td></tr>
<tr><td>Binary Image</td><td colspan="12"></td></tr>
<tr><td>Strip widths</td><td>15</td><td>56</td><td>34</td><td>27</td><td>63</td><td>27</td><td>33</td><td>57</td><td>64</td><td>56</td><td>34</td><td>15</td></tr>
</table>

Assigned Bits	1	00	1	0	11	0	1	00	11	00	1	0

Figure 10: Strip width calculation

7. Results and Discussion

Figure 10 shows an example of decoding a sample message using the receiver program. Along with the various stages of the frame analysis, the strip widths and their corresponding number of assigned bits are added in the figure. It can be seen that the strip width for the same amount of bits is different for white and black strips. For this particular calibration, the single strip width for black is 33, whereas for white it is 27. For two bits, the strip width is close to double, 64 and 56, respectively.

Another thing to note here is that there may be parts of strips from the adjoining frames. The width of these strips may vary from zero to two bit strip widths. When the frames are stacked, these strips give a narrower strip with an unusual width. This is because of the shutter loss, and in this case, a fixed value is added to the width before calculating the number of bits.

Reducing the pulse time increases speed but results in a smaller strip width for the same fps. The smearing around the edges of the strips remains the same, so the percentage uncertainty increases.

Increasing the distance between the camera and the LED matrix increases smearing and interference with background light. For greater accuracy, the study was performed in a dark room at a short distance.

The receiver program uses multithreading to make the communication in real time. The program is split into two parts, where one uses image processing to extract raw binary data, and that data is passed to the second part, which contains various decoding algorithm, such as Manchester, NRZ, etc., to extract the actual data.

8. Conclusion

This project was a part of a collaborative thesis, performed by Saikat Chakraborty and Monishanto Biswas, Department of Electrical and Electronic Engineering, Khulna University of Engineering and Technology, Khulna. The goal was to design and develop a novel communication method that relies on the commercially available materials to push the limits of visible light communication. The project also encompasses the knowledge attained from the journey through the previous courses of the Electrical and Electronic Engineering discipline.

There is a lot of room for future improvements in this project. For example, a short video has to be prepared for the purpose of calibration in the beginning of every new configuration of transmission. A new version of the receiver software should have the auto calibration feature, where it uses some technique like a moving average to dynamically calibrate the strip width and shutter loss values.

Currently, the range of applications of this communication system is very narrow. The experiment was performed in a dark room with a very small distance between the camera and the light source. But this range can be increased if the light can be guided, i.e., Optical fiber. Or light-limiting cylinders at both transmission and receiving end that prevents background radiation from leaking into and causing smearing.

Visible light communication can be a very reliable method of transmitting data since it is free from high-energy electromagnetic wave interference. Although there are various forms of VLC is available, but the biggest achievement of this project is implementing this scheme by finding alternative usage of the commercially available materials at reduced cost.

9. Reference

[1] http://www.pololu.com/product/1698

[2] https://www.logitech.com/en-eu/shop/p/c922-pro-stream-webcam

[3] https://en.wikipedia.org/wiki/Breadboard

[*] https://www.tinkercad.com

[*] https://www.create.arduino.cc/projecthub