Li-Fi Data Transmission Using LED Blinking via PSoC and Image-Capturing Camera Receiver

T2-22-23-VL 864 / Embedded Systems Design - Embedded Project 4

Kavya Agarwal
dept. ECE
IIIT Bangalore
MT2022507
Kavya.Agarwal@iiitb.ac.in

Ritesh Lalwani
dept. ECE
IIIT Bangalore
MT2022516
Ritesh.Lalwani@iiitb.ac.in

Anwit Damale
dept. ECE
IIIT Bangalore
IMT2020532
Anwit.Damale@iiitb.ac.in

Vamsidhar Reddy
dept. ECE
IIIT Bangalore
IMT2020541
Vamsidhar.Paluchani@iiitb.ac.in

Kurian Polachan
Assistant Professor
IIIT Bangalore
Kurian.Polachan@iiitb.ac.in

***Abstract*—This study explores the possibility of using LED lights as a means of transmitting data via Li-Fi technology. The data is captured by an Android smartphone camera, while the LED transmitter is implemented through PSoC-4 using the UART protocol for low-frequency communication. The camera captures the subtle changes in the state of the LED light, which are not visible to the human eye, and these images are used to determine the presence or absence of the ON/OFF condition. With this method, data can be decoded and presented effectively. In addition, it should be noted that the data being transferred through this method is in binary format, and the transmission occurs at high frequencies. Although the changes in the LED light state are imperceptible to the human eye, the camera can detect them, allowing for effective data transfer.**

***Index Terms*—Li-fi Technology, UART Protocol, PSoC-4, Low-frequency communication**

I. Introduction

In recent years, Li-Fi technology has emerged as a promising alternative to traditional Wi-Fi for data transmission. It uses visible light communication (VLC) to transmit data through LED lights, making it a highly secure and efficient way of transmitting information.

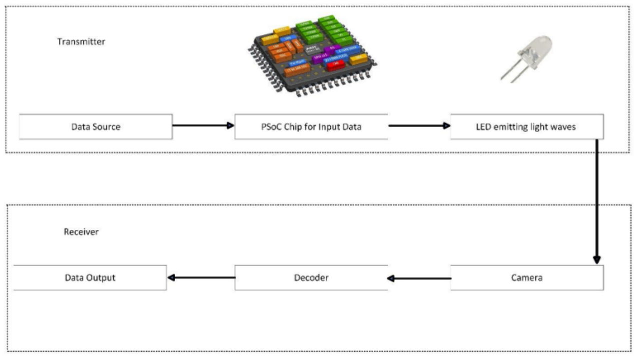


Fig. 1: Level 1 Diagram having Transmitter and Receiver Side.

In this study, we investigate the use of LED light to act as a Li-Fi source in order to transmit data. Our objective is to capture data using the camera of an Android smartphone by blinking an LED. The implementation of the LED transmitter is done in PSoC-4 via UART protocol for low-frequency communication. The camera of the smartphone is used as a receiver in order to capture the continuous changes in the state (on-off) of the light, which is invisible to the human eye. The camera captures the images which are then used to retrieve the presence of LED ON/OFF condition. Thus, the decoding of data can be done and can be presented. This study aims to provide a novel approach to data transmission through Li-Fi technology, which could have potential applications in various fields such as healthcare, military, and aerospace.

II. Methodology

- LED Transmitter Implementation using UART.
- Capturing Video through Camera.
- Extracting Image through Camera.
- Decoding LED ON/OFF condition using OpenCV.
- Decoding UART Data using RIPYL library.

A. LED Transmitter Implementation using UART

This subsection involves setting up the PSoC-4 microcontroller to control the LED, configuring UART protocol for data transmission, and controlling the LED to transmit binary data. The UART protocol is used for low-frequency communication, which provides a reliable method for data transmission.

Working, Schematics and Codes:

- In order to send the known data in a loop, the Python code continuously sends the 8-bit data packet containing the

value '0x0A' through the UART communication channel. This data is then received by the PSoC microcontroller and the LED is controlled accordingly.
When the UART receiver detects the start bit of the data packet, it begins to receive the data in sequence, starting with the least significant bit (LSB) first. As each bit is received, it is checked to determine whether it is a '0' or '1'. If the bit is a '0', the LED is turned on, and if it is a '1', the LED is turned off.
Once all 8 bits have been received and processed, the UART receiver waits for the next start bit to begin receiving the next packet of data. This process repeats continuously, resulting in the LED flashing on and off in response to the incoming data.

- The schematic in the PSoC Creator and configuration are as follows.

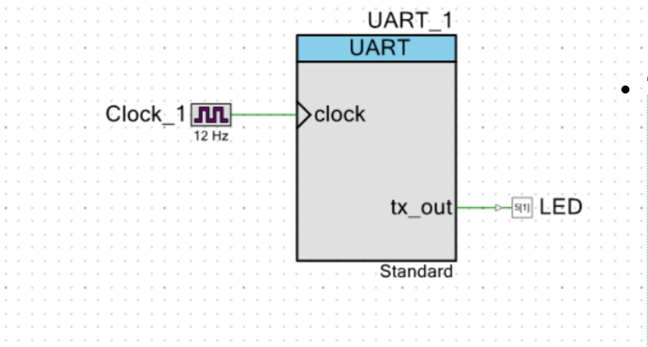
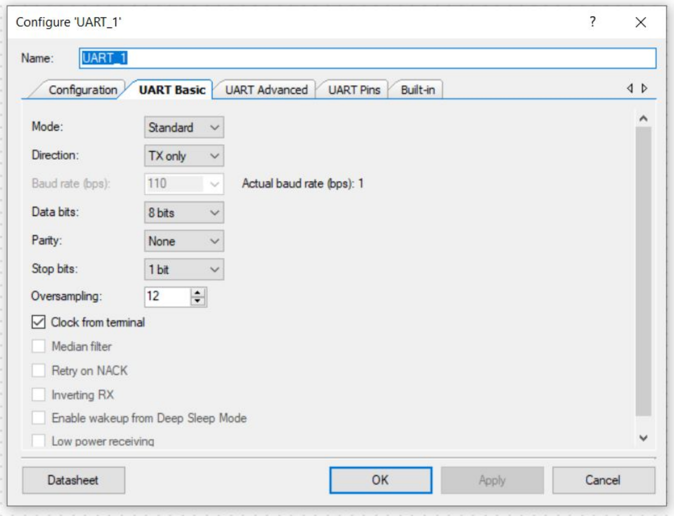


Fig. 2: Schematic of UART in the PSoC Creator.



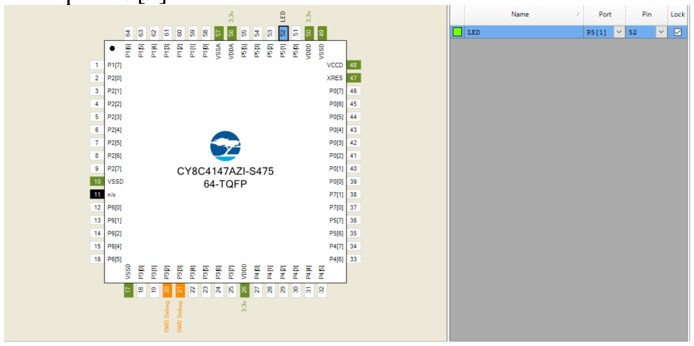
Name: UART_1

Mode: Standard
Direction: TX only
Baud rate (bps): 110 Actual baud rate (bps): 1
Data bits: 8 bits
Parity: None
Stop bits: 1 bit
Oversampling: 12
☑ Clock from terminal
☐ Median filter
☐ Retry on NACK
☐ Inverting RX
☐ Enable wakeup from Deep Sleep Mode
☐ Low power receiving

Fig. 3: UART Block Configuration in the PSoC Creator.

- The Minimum baud rate selected to drive LED is 1bps. However, we can increase the baud rate to an extent such that the camera is able to capture frames in a second accordingly and no data is lost. The LED Selected is green and with 'Strong Drive' driving mode.

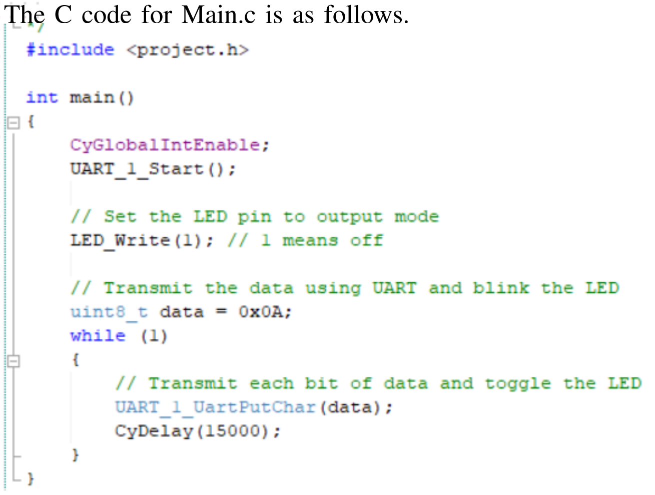
- The LED turns ON when 0 is written (as it goes to ground) as defined by port P5[2]. We have shorted the ports P5[1] and P5[2], such that the data received in P5[1] is passed to the LED pin P5[2].



	Name	Port	Pin	Lock
🟩	LED	P5[1]	52	☑

Fig. 4: Pins and Ports as chosen in the PSoC Creator.

- The C code for Main.c is as follows.



```
#include <project.h>

int main()
{
    CyGlobalIntEnable;
    UART_1_Start();

    // Set the LED pin to output mode
    LED_Write(1); // 1 means off

    // Transmit the data using UART and blink the LED
    uint8_t data = 0x0A;
    while (1)
    {
        // Transmit each bit of data and toggle the LED
        UART_1_UartPutChar(data);
        CyDelay(15000);
    }
}
```

Fig. 5: UART Main.c code in PSoC Creator.

So, this is how the LED Blinks. The blinking is according to the data transmission and the pattern may change accordingly. We are working on '0x0A' data transmission and the video is recorded using a 30fps mobile camera.

B. Capturing Video through Camera

This subsection involves setting up an Android smartphone camera to capture video of the LED blinking, adjusting the camera settings for optimal video quality, and recording the LED blinking. The captured video is essential for further analysis of LED blinking patterns. The camera used captures almost 30 frames in a second.

C. Extracting Image through Camera

In this subsection, the recorded video is processed to extract images of the LED blinking, enhance the images to increase LED contrast and isolate the LED area for further analysis. This step is crucial for detecting the presence or absence of LED blinking and for improving the accuracy of data

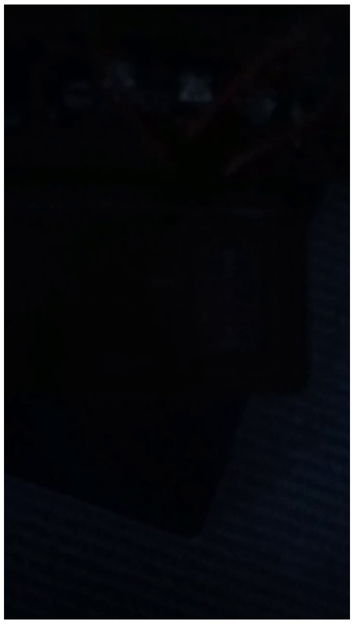


Fig. 6: This is how OFF LED looks like.

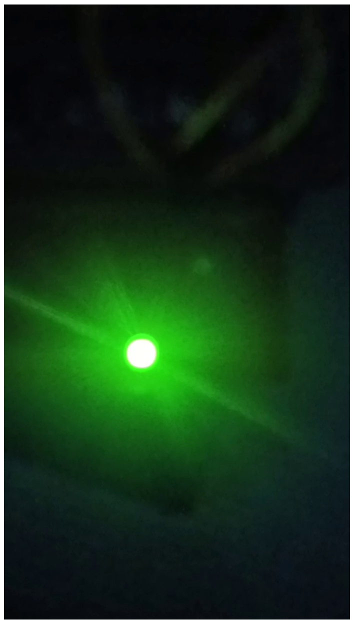


Fig. 7: This is how ON LED looks like.

decoding. This is the main step to detect the ON and OFF status of the LED to get the data.
Python codes for different steps:

- Step1 is installing the OpenCV library and reading the video using cv2.VideoCapture.

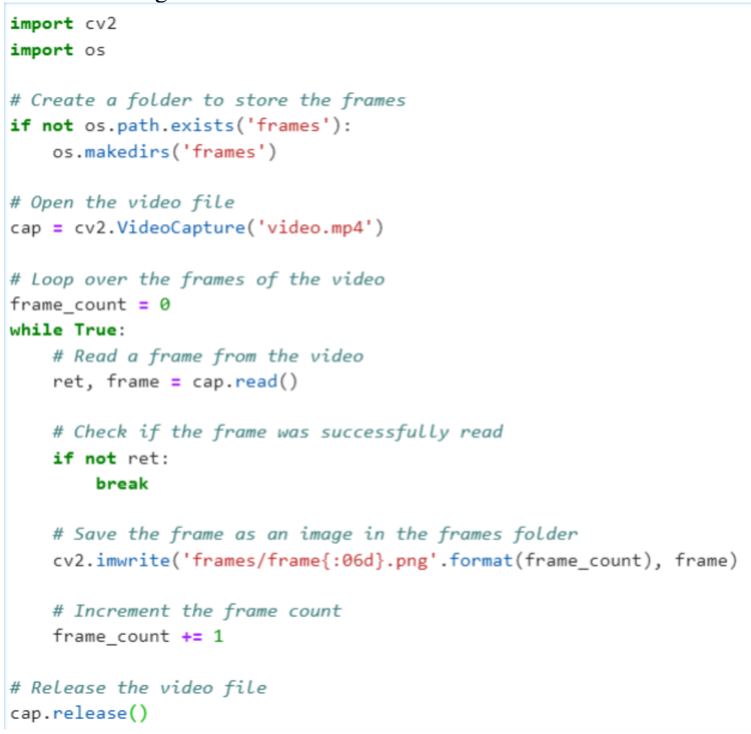
```
import cv2
cap=cv2.VideoCapture('video.mp4')
if cap:
  print('Video captured successfully!')
else:
  print('Failed to capture video!')
```

- Step2 is extracting the frames out of the video and counting the total number of frames captured.

```
y=get(cv2.CAP_PROP_FRAME_COUNT)
x = cap.y
total_frames = int(x)
total_frames
3260
```

  The count is around = 30*2*60, as the length of the video is around 2 minutes.

- Step3 is extracting all 3260 images in a folder and observing the LED ON and OFF conditions.



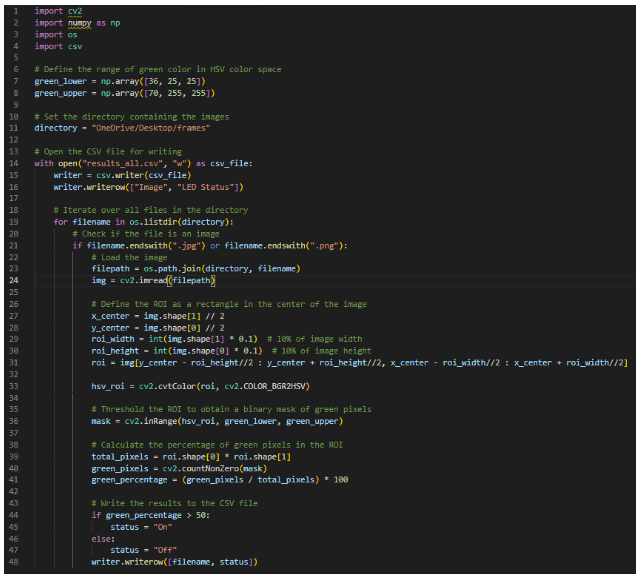
```
import cv2
import os

# Create a folder to store the frames
if not os.path.exists('frames'):
    os.makedirs('frames')

# Open the video file
cap = cv2.VideoCapture('video.mp4')

# Loop over the frames of the video
frame_count = 0
while True:
    # Read a frame from the video
    ret, frame = cap.read()

    # Check if the frame was successfully read
    if not ret:
        break

    # Save the frame as an image in the frames folder
    cv2.imwrite('frames/frame{:06d}.png'.format(frame_count), frame)

    # Increment the frame count
    frame_count += 1

# Release the video file
cap.release()
```

Fig. 8

As per the code, the LED is OFF initially and then it is ON and OFF according from LSB to MSB data for 0x0A.

D. Decoding LED ON/OFF condition using OpenCV

After the images have been processed and the LED area has been localized, the next step is to decode the binary data transmitted by the LED. This involves analyzing the pattern of blinking on and off of the LED, which corresponds to a sequence of '0' and '1' bits. To accomplish this, the binary image of the detected LED area is segmented into individual frames, each representing a single blink of the LED. The frames are then analyzed to determine the duration of each blink and the time between successive blinks. Based on this information, the sequence of '0' and '1' bits can be decoded. Once the binary data has been decoded and synchronized, it can be used for a wide range of applications, such as remote control of electronic devices, data transmission in low-power wireless sensor networks, and control of autonomous robots and drones. The ability to transmit data using only the blinking of an LED is a valuable tool in situations where other forms of wireless communication are not feasible or practical.



```python
import cv2
import numpy as np
import os
import csv

# Define the range of green color in HSV color space
green_lower = np.array([36, 25, 25])
green_upper = np.array([70, 255, 255])

# Set the directory containing the images
directory = "OneDrive/Desktop/frames"

# Open the CSV file for writing
with open("results_all.csv", "w") as csv_file:
    writer = csv.writer(csv_file)
    writer.writerow(["Image", "LED Status"])

    # Iterate over all files in the directory
    for filename in os.listdir(directory):
        # Check if the file is an image
        if filename.endswith(".jpg") or filename.endswith(".png"):
            # Load the image
            filepath = os.path.join(directory, filename)
            img = cv2.imread(filepath)

            # Define the ROI as a rectangle in the center of the image
            x_center = img.shape[1] // 2
            y_center = img.shape[0] // 2
            roi_width = int(img.shape[1] * 0.1)  # 10% of image width
            roi_height = int(img.shape[0] * 0.1)  # 10% of image height
            roi = img[y_center - roi_height//2 : y_center + roi_height//2, x_center - roi_width//2 : x_center + roi_width//2]

            hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

            # Threshold the ROI to obtain a binary mask of green pixels
            mask = cv2.inRange(hsv_roi, green_lower, green_upper)

            # Calculate the percentage of green pixels in the ROI
            total_pixels = roi.shape[0] * roi.shape[1]
            green_pixels = cv2.countNonZero(mask)
            green_percentage = (green_pixels / total_pixels) * 100

            # Write the results to the CSV file
            if green_percentage > 50:
                status = "On"
            else:
                status = "Off"
            writer.writerow([filename, status])
```

Fig. 9: Detection of LED ON/OFF Condition.

After detecting the LED ON/OFF condition, we wrote it to a CSV file for further analysis. From this file, we extracted two waveforms that represent the status of the LED and the transmitted data. These waveforms were analyzed to extract various parameters for evaluating and optimizing our system's performance. The CSV file and waveform analysis proved to be an effective approach for analyzing the LED data and can be adapted to other data transmission systems.

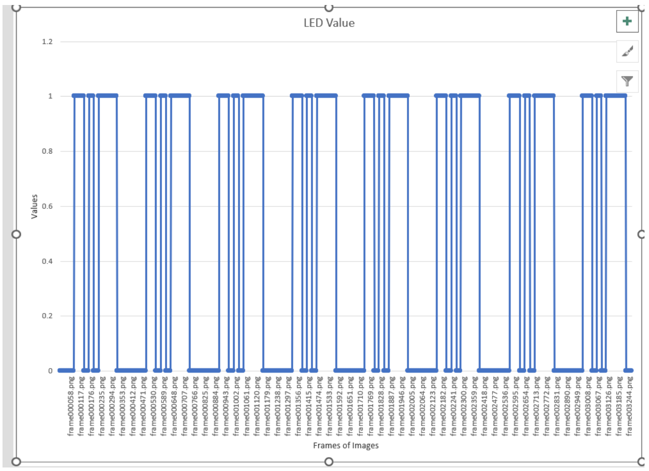


Fig. 10: Waveform of LED turning ON and OFF.

Taking LED ON condition as transmitted data '0' and OFF Condition as Trasmitted data '1' we get data waveform of data as:

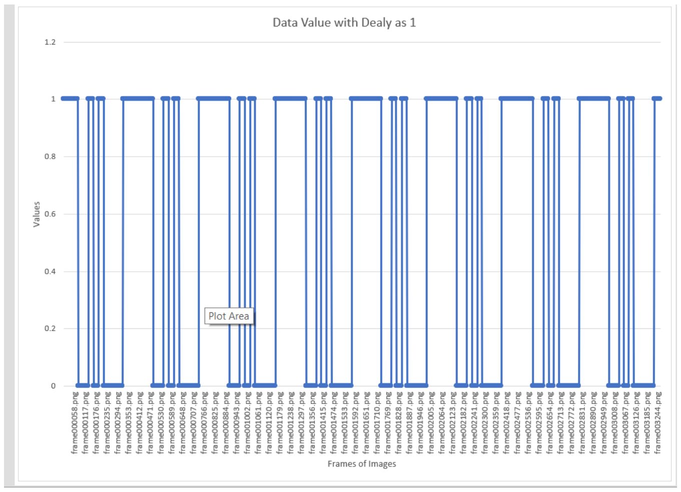
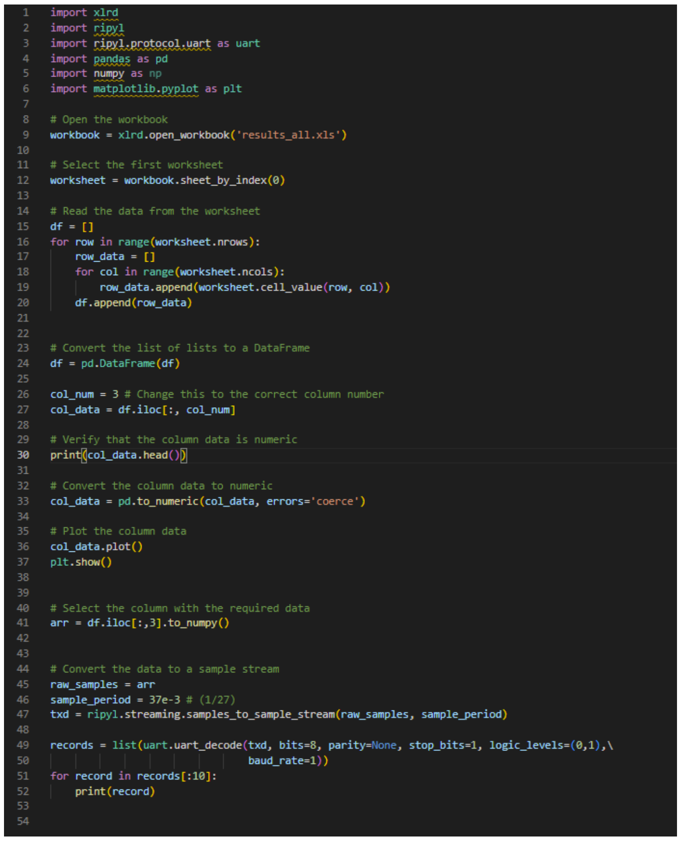


Fig. 11: Waveform of data.

E. *Decoding UART Data using RIPYL library*

This subsection involves extracting UART data from the LED control signals, decoding the binary data using UART protocol, and validating the decoded data.



```python
import xlrd
import ripyl
import ripyl.protocol.uart as uart
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Open the workbook
workbook = xlrd.open_workbook('results_all.xls')

# Select the first worksheet
worksheet = workbook.sheet_by_index(0)

# Read the data from the worksheet
df = []
for row in range(worksheet.nrows):
    row_data = []
    for col in range(worksheet.ncols):
        row_data.append(worksheet.cell_value(row, col))
    df.append(row_data)


# Convert the list of lists to a DataFrame
df = pd.DataFrame(df)

col_num = 3 # Change this to the correct column number
col_data = df.iloc[:, col_num]

# Verify that the column data is numeric
print(col_data.head())

# Convert the column data to numeric
col_data = pd.to_numeric(col_data, errors='coerce')

# Plot the column data
col_data.plot()
plt.show()


# Select the column with the required data
arr = df.iloc[:,3].to_numpy()


# Convert the data to a sample stream
raw_samples = arr
sample_period = 37e-3 # (1/27)
txd = ripyl.streaming.samples_to_sample_stream(raw_samples, sample_period)

records = list(uart.uart_decode(txd, bits=8, parity=None, stop_bits=1, logic_levels=(0,1),\
                                baud_rate=1))
for record in records[:10]:
    print(record)
```

Fig. 12: Python code in Ripyl library to decode waveform.

Once, the decoding using Python is done, we get the waveform as shown below of the decoded data.

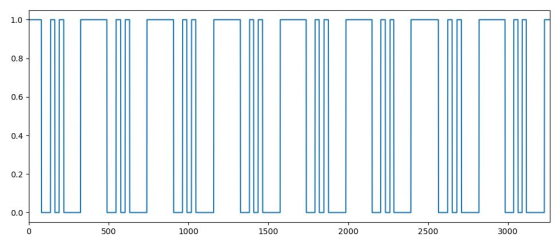


Fig. 13: Final Waveform.

III. Future Work

- Multi-LED Transmitter
- Non-binary Data Transmission

A. *Multi-LED Transmitter*

The current study uses a single LED for data transmission. Future work can explore the use of multiple LEDs to increase the data transfer rate.

B. *Non-binary Data Transmission*

The current study focuses on binary data transmission. Future work can explore the use of different encoding schemes to transmit non-binary data, such as analog data or multi-level digital data.

IV. Conclusion

This study explores the use of LED lights as a means of transmitting data via Li-Fi technology. The method involves capturing the subtle changes in the state of the LED light using an Android smartphone camera and using these images to determine the presence or absence of the ON/OFF condition. The LED transmitter is implemented through PSoC-4 CY8C4147AZI-S475 using the UART protocol for low-frequency communication. The data being transferred through this method is in binary format, and the transmission occurs at 1 bps.
The methodology of the study involves LED transmitter implementation using UART, capturing video through a camera, extracting images through a camera, decoding LED ON/OFF conditions using OpenCV, and decoding UART data using RIPYL library. The LED transmitter is controlled to transmit binary data through the UART protocol for data transmission. The camera of the smartphone captures the images, which are then used to retrieve the presence of LED ON/OFF condition, and the decoding of data can be done and presented. The study provides a promising alternative to traditional Wi-Fi for data transmission, making it a highly secure and efficient way of transmitting information.

References

[1] A. Duquel, R. Stanica, H. Rivano and A. Desportes, ”Decoding methods in LED-tosmartphone bidirectional communication for the IoT,” 2018 Global LIFI Congress (GLC), Paris, France, 2018, pp. 1-6, doi: 10.23919/GLC.2018.8319118

[2] Danakis, C., Afgani, M., Povey, G., Underwood, I., Haas, H. (2012). Using a CMOS camera sensor for visible light communication. Globecom Workshops (GC Wkshps), 2012 IEEE, 1244-1248.

[3] A. E. Marcu, R. A. Dobre and M. Vlădescu, ”Flicker Free Visible Light Communication Using Low Frame Rate Camera,” 2018 International Symposium on Fundamentals of Electrical Engineering (ISFEE), Bucharest, Romania, 2018, pp. 1-4, doi: 10.1109/ISFEE.2018.8742447.

[4] Alexis Duque, Razvan Stanica, Hervé Rivano, Adrien Desportes. Demo: Off-the-Shelf Bi-Directional Visible Light Communication Module for IoT Devices and Smartphones. EWSN 2017 - 13th International Conference on Embedded Wireless Systems and Networks, Feb 2017, Uppsala, Sweden. ACM, EWSN ’17 Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks, pp.238-239, 2017. ffhal-01473706.

[5] Sreesha Damodaran, Talal Shaikh, Prof. Nicholas K Taylor. Using Mobile Phone Based Camera to Read Information from A Li-Fi Source.