Visual Transmission

Team

- Aaditya Baranwal
- Aayush Gautam
- Muskaan Gautam

Project Overview

The project uses an Arduino board, Light Dependent Resistor (LDR), and Light Emitting Diode (LED) to transmit data using visible light. The project also uses Python programming to process image data and perform encoding and decoding of the data.

The communication system uses a modified Huffman encoding algorithm to compress the image data before transmission. The Huffman codes are generated by analyzing the image data and identifying the most frequent characters in the image data. These characters are then assigned shorter codes in the Huffman code table, while less frequent characters are assigned more extended codes.

The **TransmissionData** class initializes by creating a string from the image data and encoding it using Huffman encoding. It then sends the encoded string over the LiFi link using the serial communication protocol. The **RecievedData** class initializes by receiving the encoded string from the LiFi link, decoding it using the Huffman code table, and constructing an image from the decoded string.
LiFi technology uses visible light to transmit data. The LiFi system consists of a transmitter and a receiver. The transmitter uses an LED to transmit data by modulating the light output of the LED. The receiver uses a light sensor, such as an LDR, to detect the modulated light signal and extract the transmitted data. The modulation is typically done using on-off keying (OOK), where the LED is turned on and off at a high frequency to encode the data. The receiver can then demodulate the signal by detecting the changes in light intensity and recovering the transmitted data.

In this project, the image data is transmitted using on-off keying (OOK) modulation by controlling the light output of the LED. The LDR detects the light signal, and the data is extracted using a decoding algorithm implemented in Python. Since visible light cannot penetrate walls, LiFi technology is helpful for secure data transmission in areas where radio frequency signals may be intercepted. However, the line-of-sight nature of the transmission limits the range and reliability of the system.

CodeFlow

Python

TransmissionData and ReceivedData are used to send and receive data, respectively. Here's a brief overview of the code:
TransmissionData class:
The **__init__** method initializes the object with the given image and Huffman codes.
make_string_from_image method creates a string of characters from the given image.
huffman_encode method encodes the string using Huffman encoding.
send method sends the encoded string to the Arduino using serial communication.
gen_huffman_codes method generates Huffman codes for the images in the given directory.
ReceivedData class:
The **__init__** method initializes the object with the Huffman codes.
huffman_decode method decodes the received string using the Huffman codes.
make_image_from_string method creates an image from the decoded string.
receive method receives the encoded string from the Arduino using serial communication.
The code uses the **pyserial** library to communicate with Arduino. The **pyplot** module from the **matplotlib** library is used to show the images. Also, the collections, **heapq**, and **json** modules from Python's standard library are used for encoding and decoding the string using Huffman encoding.

Arduino

The first Arduino code file, **Receiver.ino**, sets up an LDR sensor on ***pin A0*** and waits for a change in the sensor state. When the LDR detects a change in the amount of light received (i.e., when a Li-Fi signal is detected), the code decodes the incoming data by reading the changes in light intensity as binary signals. The incoming binary signals are decoded byte-by-byte and printed out via serial communication.

The code first defines some constants like the ***LDR_PIN***, ***THRESHOLD***, and ***PERIOD*** and sets up the LDR sensor on the ***LDR_PIN*** as an input. The **get_ldr()** function reads the sensor's output voltage in the loop function. If the voltage is above the threshold, the **get_ldr()** function returns true; if not, it returns false.

The variable current_state is then set to the output of the **get_ldr()** function, and if ***current_state*** is different from ***previous_state***, the **get_byte()** function is called to decode the incoming binary signal. The **get_byte()** function reads the incoming signal byte by byte and returns the decoded byte, which is then printed out by the **print_byte()** function.

The second Arduino code file, **Transmitter.ino**, sets up an LED on ***pin 3*** and sends a pre-defined string as Li-Fi signals to the receiver. The LED is set up as an output in the setup function, and the serial communication is initialized. In the loop function, the **send_byte()** function is called for each character in the string. The send_byte() function sends each byte of data by first turning the LED off for a fixed period, then sending the byte bit by bit by turning the LED on and off, with the duration of each LED state controlled by the ***PERIOD*** constant.

Takeaway

The essential components used in this project are the LDR sensor and the LED. LDRs are commonly used as light sensors, and in this project, the LDR is used to detect changes in light intensity caused by the Li-Fi signal. The LED is the transmitter to encode the data into binary signals by modulating the LED's light intensity. The Li-Fi technology used in this project is a form of optical wireless communication that uses visible light to transmit data. Li-Fi has the potential to

provide high-speed, secure, and low-latency wireless communication and has been proposed as an alternative to traditional Wi-Fi and cellular networks.