Project Name: Simulating LiFi Communication

Project Participants: B020, B035, B036, B042

Introduction:

String transmission through LiFi using Arduino and LED and Light Dependent Resistors (LDRs) is a project that aims to utilize the benefits of LiFi technology to achieve secure, high-speed data transfer rates between two devices. The project involves the use of an Arduino board, LEDs, and LDRs to transmit data through visible light communication. Background LiFi technology is a type of wireless communication technology that uses light instead of radio waves to transmit data. This technology is based on visible light communication, which involves the modulation of light intensity to transfer binary data. LiFi has several advantages over traditional WiFi, including higher data transfer rates, greater security, and less interference with other electronic devices. LiFi technology is still in its early stages of development, but it has the potential to revolutionize the way we transmit and access data.

Objectives:

The main objective of this project is to demonstrate the feasibility of string transmission through LiFi using Arduino and LED and LDRs. The specific objectives of the project are:

1. To design and build a LiFi transmitter and receiver system using Arduino boards, LEDs, and LDRs.
2. To test the system to determine its data transfer rate and its ability to transmit data securely.
3. To compare the performance of the LiFi system with that of traditional WiFi.

Apparatus:

- 1 x ESP 32
- 1 x Arduino Uno
- 1 x Breadboard
- 1 x LDR
- 1 x LED
- Wires

Working:

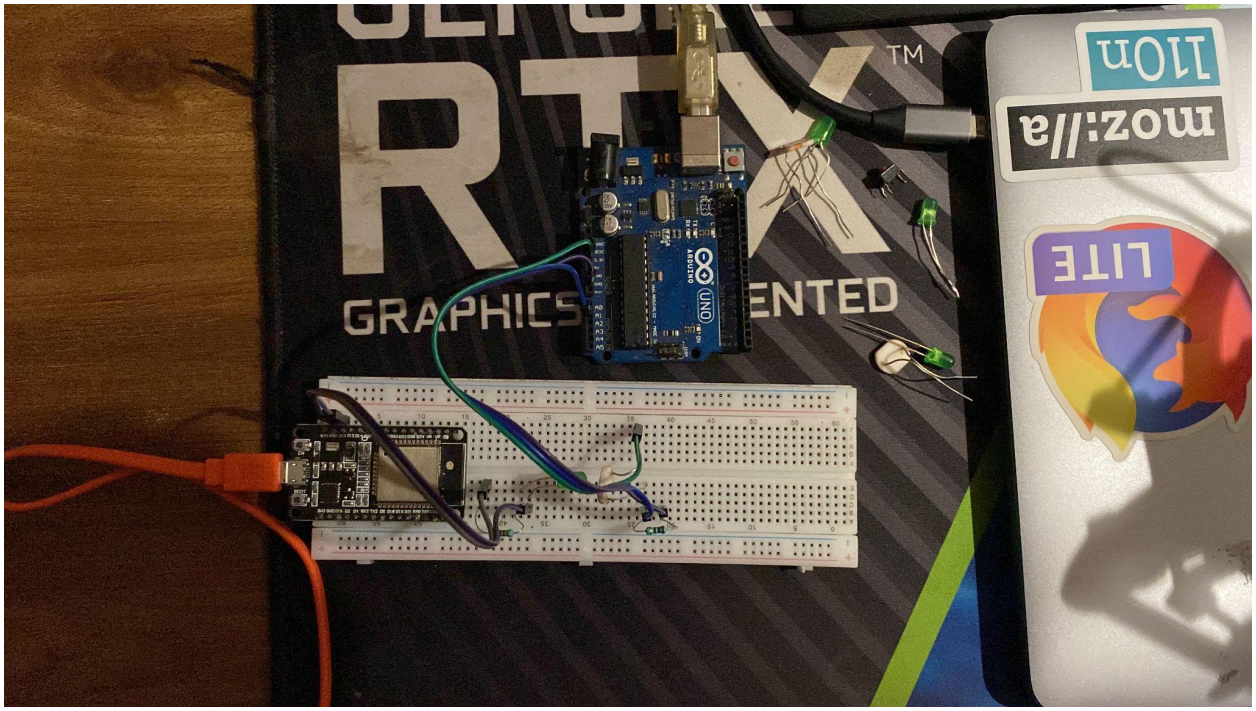
Light pulsing is one of the key methods used to transmit data using LiFi technology. In LiFi, data is transmitted by modulating the intensity of the light emitted by an LED bulb, which is then received by a photodetector, such as a Light Dependent Resistor (LDR). The modulation of light intensity is achieved by rapidly turning the LED on and off, creating a sequence of light pulses that represent binary data.

To transmit data, the LED is switched on and off at a very high frequency, typically several megahertz, which is much faster than the human eye can perceive. The frequency at which the LED is switched on and off determines the data rate, with higher frequencies corresponding to higher data rates. Each pulse of light represents a single bit of data, with a sequence of pulses representing a sequence of bits that make up the transmitted data.

At the receiver end, the LDR detects the pulses of light and converts them back into binary data. The LDR is sensitive to changes in light intensity, allowing it to detect the modulated light pulses even in the presence of ambient light. The detected pulses are then processed to recover the original binary data transmitted by the LED.

Overall, light pulsing is a highly efficient and secure method of transmitting data using LiFi technology. It is capable of achieving high data rates with minimal interference and provides a secure alternative to traditional WiFi, making it an attractive option for a wide range of applications.

Hardware:



Code (Transmission):

```
#include <Arduino.h>

const int ledPin = 13;
const int bitRate = 100;
const String startBits = "00000010";
const String endBits = "00000011";

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);
}

void loop() {
  String message = "Anish Nair GTA";     // Replace with the message you
want to send.
  String binaryMessage = startBits+encodeToUTF8(message)+endBits;

  Serial.println("Transmitting message: " + message);
  Serial.println("In binary format with prefix and suffix bits: " +
binaryMessage);

  sendBinaryMessage(binaryMessage);
  delay(1000);
}

void sendBinaryMessage(String binaryMessage) {
  for (int i = 0; i < binaryMessage.length(); i++) {
    digitalWrite(ledPin, binaryMessage[i] == '1' ? HIGH : LOW);
    delay(bitRate);
  }
}

#include <string.h>

// Function to encode text to UTF8 to 1s and 0s
String encodeToUTF8(String input) {
  String output = "";
  int len = input.length();
```

```
  for (int i = 0; i < len; i++) {
    byte c = input.charAt(i);
    if (c < 128) {
      output += "0";
      for (int j = 6; j >= 0; j--) {
        output += bitRead(c, j);
      }
    } else if ((c > 191) && (c < 224)) {
      byte c2 = input.charAt(++i);
      output += "110";
      output += ((c >> 2) & 0x07);
      output += "10";
      output += ((c2 >> 6) & 0x03);
      output += ((c2 >> 0) & 0x3F);
    } else {
      byte c2 = input.charAt(++i);
      byte c3 = input.charAt(++i);
      output += "1110";
      output += ((c >> 4) & 0x0F);
      output += "10";
      output += ((c2 >> 6) & 0x03);
      output += ((c2 >> 0) & 0x3F);
      output += "10";
      output += ((c3 >> 0) & 0x3F);
    }
  }
  return output;
}
```

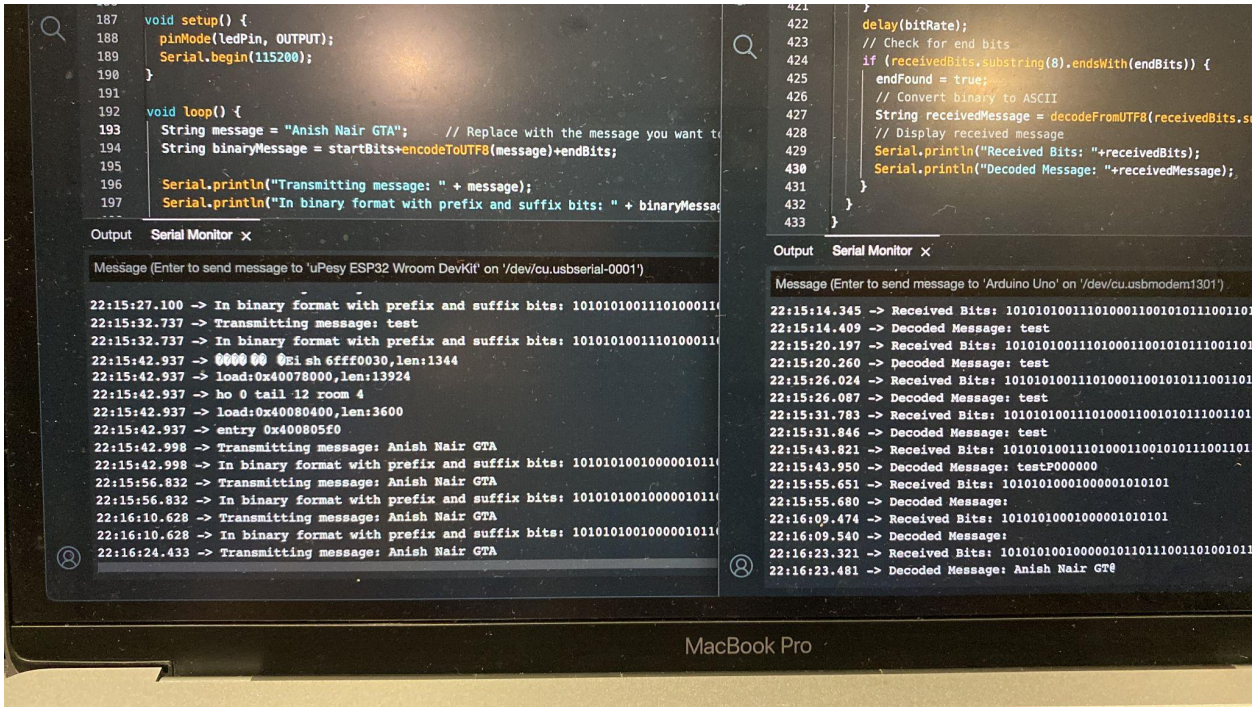
Code (Receiver):

```
#include <Arduino.h>

const int photoResistorPin = A1; // Use A0 pin for Arduino Uno
const int threshold = 256;
const int bitRate = 100;
const String startBits = "00000010";
const String endBits = "00000011";

void setup() {
  pinMode(photoResistorPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Wait for start bits
  bool startFound = false;
  String receivedBits = "";
  while (!startFound) {
    int reading = analogRead(photoResistorPin);
    // Serial.println(analogRead(photoResistorPin));
    // Check for start bits
    if (reading > threshold) {
      receivedBits += "1";
    } else {
      receivedBits += "0";
    }
    delay(bitRate);

    if (receivedBits.length() > 8) {
      receivedBits = receivedBits.substring(receivedBits.length() -
startBits.length(), receivedBits.length());
    }
    if (receivedBits == startBits)
    {
      startFound = true;
    }
  }
  // Read message bits
```

```
  bool endFound = false;
  while (!endFound) {
    int reading = analogRead(photoResistorPin);
    if (reading > threshold) {
      receivedBits += "1";
    } else {
      receivedBits += "0";
    }
    delay(bitRate);
    // Check for end bits
    if (receivedBits.substring(8).endsWith(endBits)) {
      endFound = true;
      // Convert binary to ASCII
      String receivedMessage =
decodeFromUTF8(receivedBits.substring(startBits.length(),
receivedBits.length()-endBits.length()));
      // Display received message
      Serial.println("Received Bits: "+receivedBits);
      Serial.println("Decoded Message: "+receivedMessage);
    }
  }
}


#include <string.h>

// Function to decode text from UTF8 to 1s and 0s
String decodeFromUTF8(String input) {
  String output = "";
  int len = input.length();

  for (int i = 0; i < len; i += 8) {
    byte c = 0;
    for (int j = 0; j < 8; j++) {
      if (input.charAt(i+j) == '1') {
        c |= (1 << (7-j));
      }
    }
    output += (char)c;
  }
```

```
  return output;
}
```

Output:



Limitations:

Although LiFi has several advantages over traditional radio frequency (RF) wireless communication technologies, such as higher bandwidth and lower interference, It also has some limitations, such as:

1) Limited range: LiFi has a restricted transmission range and can only send data over very close distances. The range is restricted because of the intensity of the light source, and it is possible for the signal to be obstructed by things like walls and pieces of furniture.

2) Line-of-sight communication: In order for LiFi to function properly, there must be nothing but clear sight between the transmitter and the receiver. It is possible that the data transfer will be disrupted if there is something in the path of the LED and the LDR.

3) Sensitivity to ambient light: An LDR is a light-dependent resistor, and the LDR's sensitivity to the ambient light can have a negative impact on LiFi performance. The light around could interfere with the signal and cause errors in the way the data is sent.

4) Limited Hardware Support: Because LiFi technology is still fairly new, not all pieces of hardware are compatible with it. Because the ESP-32 and the Arduino Uno do not have built-in support for LiFi, you might need to acquire additional hardware and software in order to make use of it.

5) Complexity of implementation: To successfully set up LiFi, you need to know a certain amount about electronics and programming. For beginners, making the connection between the LED and LDR and the ESP32 or Arduino Uno might be difficult, and resolving any problems that arise can take a significant amount of effort.

Conclusion:

The project shows how Arduino, LED bulbs, and LDRs can be used to successfully send data through strings using LiFi technology. The experiment shows that LiFi technology can be used as a viable alternative to traditional WiFi in situations where data security is a high priority. The results of the experiment provide a foundation for further research and development of LiFi technology for practical applications.