# Li-Fi based Text Communication Application

*Figure 1 Li-Fi Technology*

ICT 305 2.0 Embedded Systems

AS2019925

KTPR Kariyawasam

# Contents

## 1.1 What is LI-FI Technology?

Li-Fi is a wireless communication technology which utilizes light to transmit data and position between devices.

Li-Fi is a light communication system that is capable of transmitting data at high speeds over the visible light, ultraviolet, and infrared spectrums. In its present state, only LED lamps can be used for the transmission of data in visible light.

## 1.2 Introduction

Wireless communications, such as Wi-Fi and Bluetooth, are commonly used in the daily life. However, while more and more wireless devices occupy the available frequencies, the frequency spectrum is reaching a maximum capacity. Thus, Li-Fi, communicating using light waves between two devices, becomes a future solution to transmit data.

## 1.3 System Overview

This project used two different microcontrollers that is connected to two different computers, each are connected to their own serial monitor using application which is written in C#. One computer act as a data transmitter, while another computer act as the data receiver. Data between two pc are transmitted using light that is send by Nodemcu's transmitter using an LED and is captured by Nodemcu’s receiver using a LDR.

LED devices with a microcontroller may keep the lighting functionality of a LED while sending data in the background.  On the receiver side, a small LDR (Light Emitting Diode) receives the signals from LEDs. The system consists of two microcontroller devices as the emitter and receiver.

The data transmitted is string of characters that will be sent from the transmitter computer's application into receiver computer's application. This is called Bi-directional data communication between two microcontrollers.

The user interface, based on the C# programming language, is used to control sending and receiving data. LED signals the beginning and end of the data transmission During the transmission, LEDs switching on and off signal binary 0’s and 1’s.

## 2.1 Hardware Design

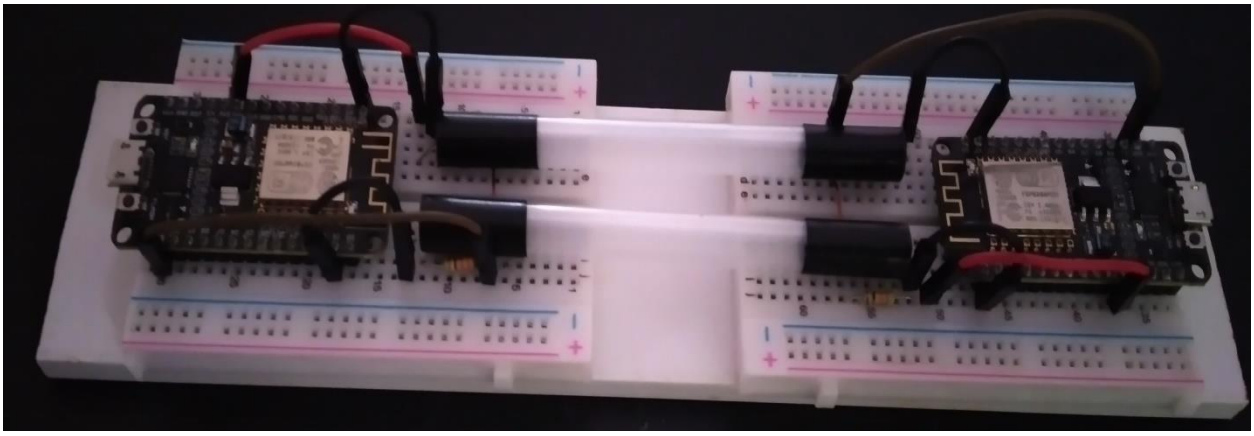

*Figure 2 Hardware design*

## 2.2 Data flow between hardware components

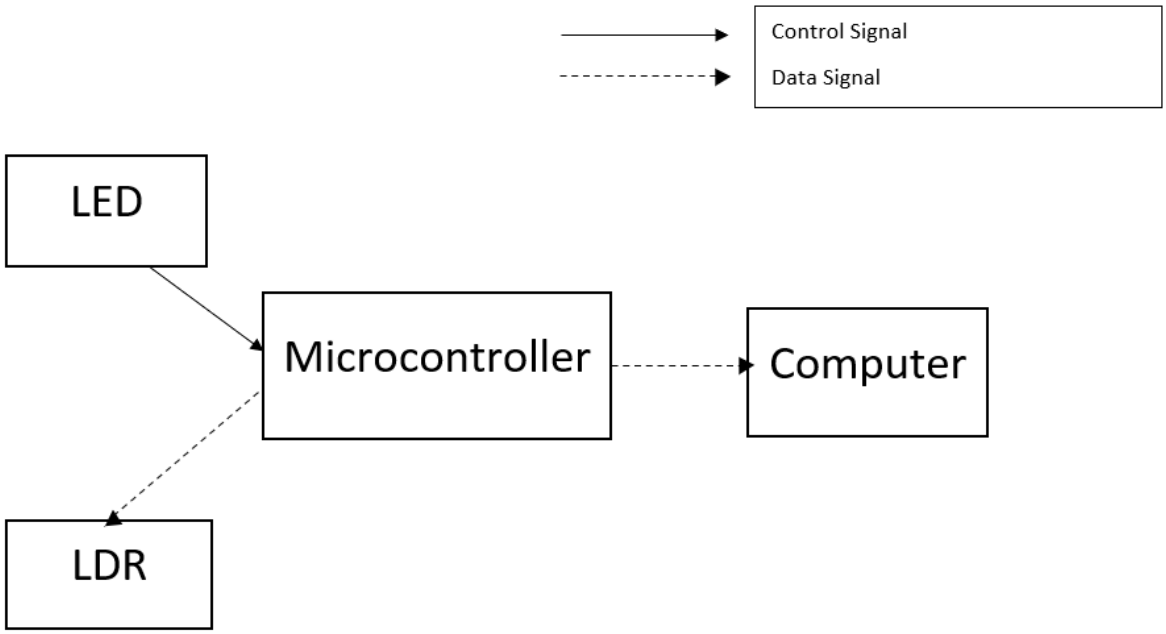


*Figure 3 Data flow between hardware components*

## 2.3 Select hardware components

### 2.3.1 LDR (Light Dependent Resistor)

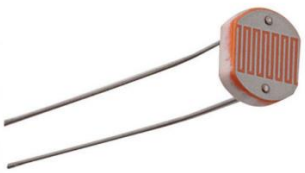

*Figure 4 LDR*

LDR is commonly used to measure the light intensity. Its resistor is decreased as the intensity of light that LDR is exposed to increase and vice versa.

### 2.3.2 LED (Light Emitting Diode)

*Figure 5 LED*

(LED) is a semiconductor device that emits light when current flows through it. The transmitter part contains a light source, which is LED that is used to transmit data.

### 2.3.3 Node MCU (esp 8266)

*Figure 6 esp 8266*

An ESP8266 Wi-Fi module is a SOC microchip mainly used for the development of endpoint IoT (Internet of things) applications. It lacks only a DC power jack and works with a Mini-B USB cable.

# 2.Circuit Diagram

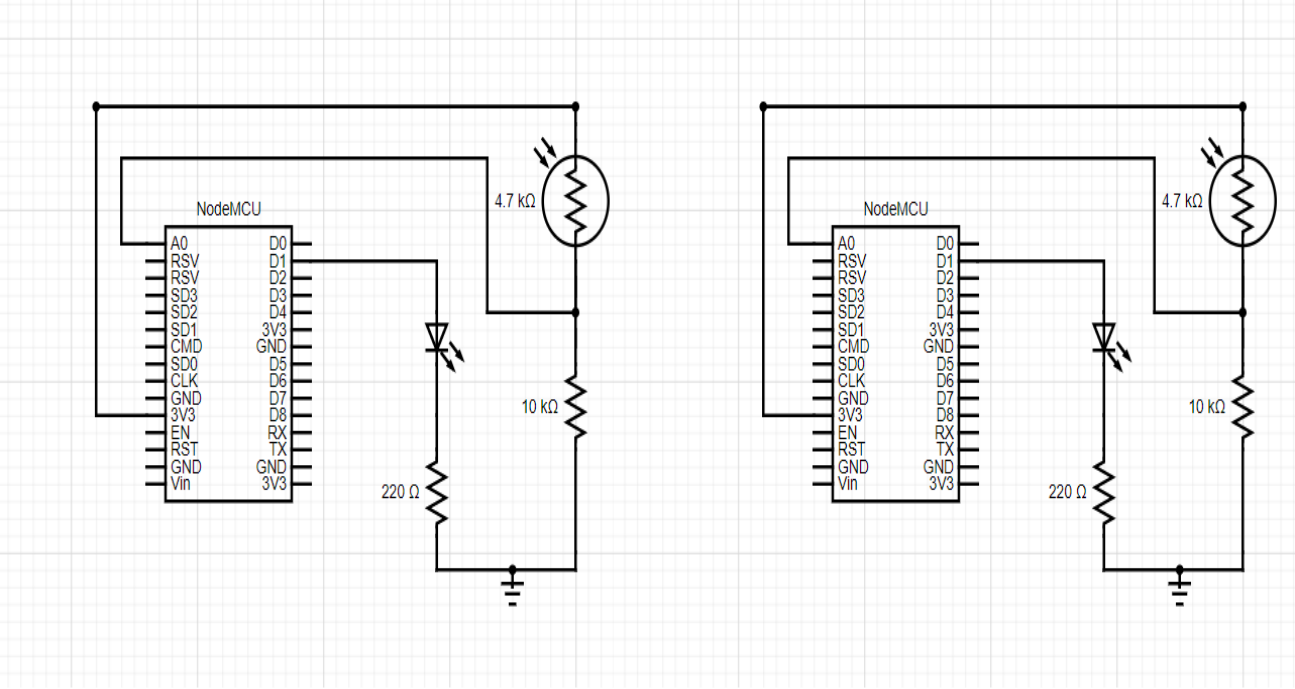


*Figure 7 Circuit Diagram*

# 2.5 Software Design

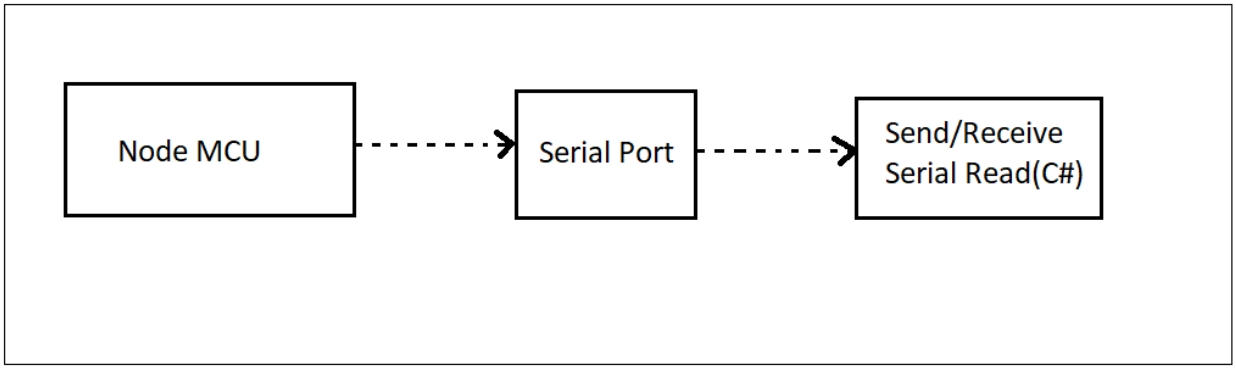


*Figure 8 Software architecture design*

### 2.5.1 User interface

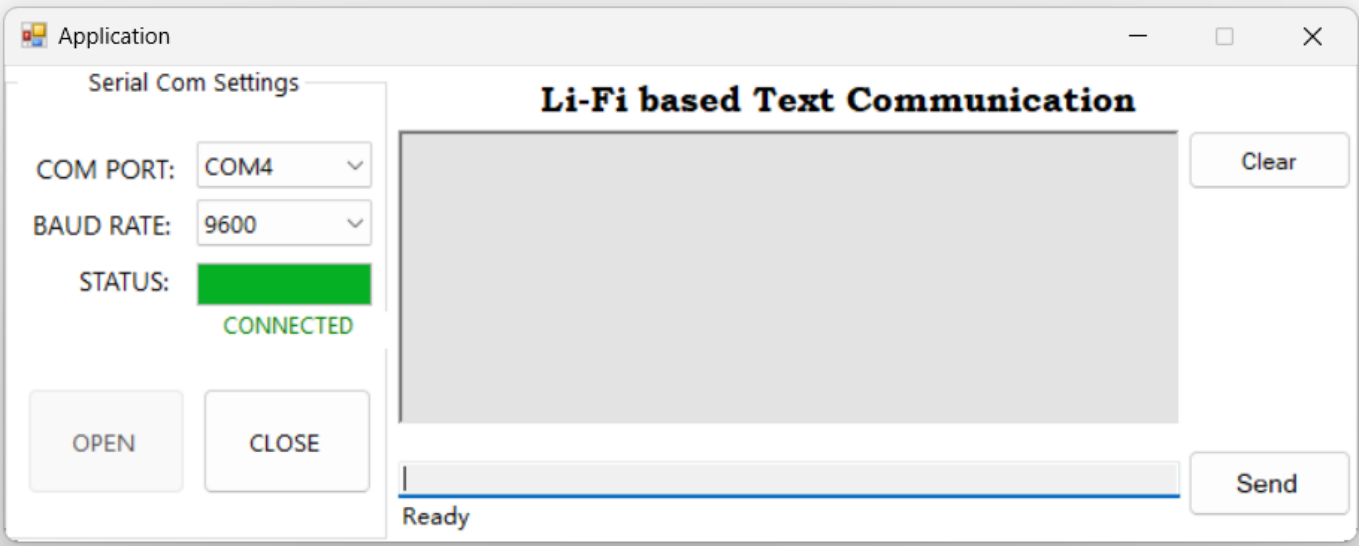


*Figure 9 Software User Interface*

## 3.1 Conclusion

The purpose of this project is to develop a potential alternative wireless connection for Wi-Fi. In the process of this project, a two-sided transmission system can transmit texts successfully with Li-Fi technique.

The transmission rate is hard to be upgraded. On the emitter side, we can easily control the LED blink rate up to 1MHz rate.

However, on the receiver side, based on the details from the internet, the discharge needs  at least 10ms to complete. A better solar diode which has higher response time may solve this problem.

## 3.2 Future Scope

Increase the transmission rate: Using high-end solar diodes which have fast reaction rate of light.

- Because this system can't transmit files such as images, documents. However, with the limitation of the LDR's response time, it is hard to reach high speeds in data transmission. With high-end and low-response-time components, it will be able to transmit files at high speeds.

## 4.1 Nodemcu code

```
 // Pin for transmitting light through LED
const int data = D1;
const int ldr = A0;

void send_data(String msg);
void receive();

// String to be sent
String myText ;
int ascii;

String t = "00000000";
String br = "";
int val;
int THRESHOLD = 800;
int x, y, te, j, k, temp = 0;
int m = 7;
void setup()
{

  pinMode(ldr, INPUT);
  pinMode(data, OUTPUT);
  Serial.begin(9600);
  delay(1000);
}

void loop()
```

```
{
  receive();
  if (Serial.available())
  {
    myText = Serial.readString();
    send_data(" DEVICE: ");
    send_data(myText);
    send_data("\n");

    Serial.print("ME: ");
    Serial.println(myText);

    Serial.flush();

  }
  else
  {
    digitalWrite(data, HIGH);  //idel status

  }
}

void send_data(String msg)
{
  for (int i = 0; i < msg.length(); i++)
  {
    ascii = msg[i];
    digitalWrite(data, HIGH);
    delay(100);
    digitalWrite(data, LOW);
    delay(100);
    digitalWrite(data, LOW);
    delay(100);
    digitalWrite(data, HIGH);
    delay(100);
    digitalWrite(data, HIGH);
    delay(100);
    digitalWrite(data, LOW);
    delay(100);
    digitalWrite(data, LOW);
    delay(100);
    digitalWrite(data, HIGH);
    delay(100);
    for (int j1 = 7; j1 >= 0; j1--)
    {
```

```
      if (bitRead(ascii, j1) == 1)
      {
        digitalWrite(data, HIGH);
        delay(100);
        digitalWrite(data, LOW);
        delay(100);

      }
      else
      {
        digitalWrite(data, LOW);
        delay(100);
        digitalWrite(data, HIGH);
        delay(100);

      }
    }

  }


}

void receive()
{
  val = analogRead(ldr);

  k++;
  temp += val;
  if (val > THRESHOLD)
    t = t.substring(1) + "1";
  else
    t = t.substring(1) + "0";
  if (t.equals("10011001"))
  {
    y = 0;
    for (int i = 15; i >= 0; i--)
    {
      delay(100);
      val = analogRead(ldr);
      k++;
      temp += val;
      if (val > THRESHOLD)
        br = br + "1";
      else
```

```
        br = br + "0";

      if (br.equals("01"))
      {
        x = 0;
        br = "";

        y += 0;
        m--;
      }
      else if (br.equals("10"))
      {
        x = 1;
        br = "";
        j = m;
        te = 1;
        while (j--)
          te = te * 2;
        y += x * te;
        m--;
      }
      else if (br.length() > 2)
        br = "";

    }
    m = 7;
    t = "00000000";
    Serial.print((char)y);

  }
  if (k >= 24)
  {
    THRESHOLD = temp / k;   //get average threshold
    k = 0;

    temp = 0;

  }
  delay(100);
}
```

## 4.2 Chat Application (C#)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.Drawing.Text;

namespace RxTx_with_Arduino
{
    public partial class SerialCom : Form
    {
        string serialDataIn;  //To get serial data
        bool waitForEnd = false;

        public SerialCom()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }
```

```
        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void SerialCom_Load(object sender, EventArgs e)
        {

            buttonOPEN.Enabled = true;
            button_CLOSE.Enabled = false;
            button_SEND.Enabled = false;
            progressBar_STATUSbar.Value = 0;
            labelSTATUS.Text = "Disconnected";
            labelSTATUS.ForeColor = Color.Red;

            comboBox_BAUD_RATE.Text = "9600";
            string[] portLists = SerialPort.GetPortNames();
            comboBox_COM_PORT.Items.AddRange(portLists);



        }

        private void buttonOPEN_Click(object sender, EventArgs e)
        {
            try
            {
                serialPort1.PortName = comboBox_COM_PORT.Text;
                serialPort1.BaudRate = Convert.ToInt32(comboBox_BAUD_RATE.Text);
                serialPort1.Open();

                buttonOPEN.Enabled = false;
                button_CLOSE.Enabled = true;
                button_SEND.Enabled = true;
                progressBar_STATUSbar.Value = 100;
                labelSTATUS.Text = "CONNECTED";
                labelSTATUS.ForeColor = Color.Green;

                comboBox_BAUD_RATE.Text= "9600";



            }
            catch (Exception error)
```

```
            {
                MessageBox.Show(error.Message);
            }

        }

        private void button_CLOSE_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                try
                {
                    serialPort1.Close();

                    buttonOPEN.Enabled = true;
                    button_CLOSE.Enabled = false;
                    button_SEND.Enabled = false;
                    progressBar_STATUSbar.Value = 0;
                    labelSTATUS.Text = "Disconnected";
                    richTextBox_Txt_Receiver.Text = "";
                    labelSTATUS.ForeColor = Color.Red;

                    comboBox_BAUD_RATE.Text = "9600";

                    sending_Status.Text = "Ready";
                    sending_Status.ForeColor = Color.Gray;
                }
                catch(Exception error)
                {
                    MessageBox.Show(error.Message);
                }
            }
        }

        private void SerialCom_FormClosed(object sender, FormClosedEventArgs e)
        {

        }

        private void SerialCom_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                try
                {
```

```
                serialPort1.Close();
            }
            catch(Exception error)
            {
                MessageBox.Show(error.Message);
            }
        }
    }

    private void button_SEND_Click(object sender, EventArgs e)
    {
        try
        {



            if (waitForEnd == true)
            {
                button_SEND.Enabled = false;

                MessageBox.Show("Please Wait!"+serialDataIn+"\n"+waitForEnd);
            }
            else
            {
                serialPort1.Write(textBox_Txt_Send.Text);
                richTextBox_Txt_Receiver.Text += "Me: ";


                textBox_Txt_Send.Text = "";
                sending_Status.Text = "Message is Sending...";
                sending_Status.ForeColor = Color.Black;
                button_SEND.Enabled = false;


            }







        }
        catch(Exception error)
        {
            MessageBox.Show(error.Message);
        }
```

```
        }

        private void serialPort1_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            serialDataIn = serialPort1.ReadExisting();



            this.Invoke(new EventHandler(ShowData));



        }

        private void ShowData(object sender, EventArgs e)

        {



            if (sending_Status.Text == "Message is Sending...")
            {
                sendingStatus();
            }
            if (sending_Status.Text == "Ready")
            {
                receivingStatusAsync();


            }







            richTextBox_Txt_Receiver.Text += serialDataIn;
        }
        private async void sendingStatus()
        {

            if (serialDataIn.Contains('\n'))
            {
```

```
                button_SEND.Enabled = true;
                sending_Status.Text = "Message Sent";
                //richTextBox_Txt_Receiver.Text += "\n";
                sending_Status.ForeColor = Color.Black;
                await Task.Delay(2000);
                sending_Status.Text = "Ready";
                sending_Status.ForeColor = Color.Black;

                waitForEnd = false;




            }


        }

        private async Task receivingStatusAsync()
        {
            if (serialDataIn == "\n")
            {

                button_SEND.Enabled = true;
                sending_Status.Text = "Message Received";

                sending_Status.ForeColor = Color.Black;
               await Task.Delay(2000);
                sending_Status.Text = "Ready";
                sending_Status.ForeColor = Color.Black;

                waitForEnd = false;




            }


        }

        private void richTextBox_Txt_Receiver_TextChanged(object sender,
EventArgs e)
        {
```

```
            richTextBox_Txt_Receiver.SelectionStart =
richTextBox_Txt_Receiver.Text.Length;
            richTextBox_Txt_Receiver.ScrollToCaret();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            richTextBox_Txt_Receiver.Text = "";
        }


    }
}
```

# 5 References


- Wikipedia. Li-Fi technology *https://en.wikipedia.org/wiki/Li-Fi*
- Researchgate.net Transfer Data from Based on Li-Fi *https://www.researchgate.net*
- Arduino projects *https://create.arduino.cc/projecthub*