INVESTIGATIVE ANALYSIS OF Li-Fi EFFICIENCY IN 2D AND 3D IMAGE TRANSMISSION

Supported by tetfund TERTIARY EDUCATION TRUST FUND

B. O. Sadiq[1]*, A. T. Salawudeen[1], H. Abdullahi[1], O. S. Zakariyya[2] and A. S. Abubakar[3]
[1]Department of Computer Engineering, Ahmadu Bello University Zaria, Nigeria
[2]Department of Electrical Engineering, University of Ilorin, Kwara, Nigeria
[3]Department of Electrical Engineering, Ahmadu Bello University Zaria, Nigeria
*Corresponding author: bosadiq@abu.edu.ng

Received: April 12, 2018 **Accepted:** October 20, 2018

Abstract: The use of wireless and networked devices is rising exponentially, making the current radio frequency spectrum overcrowded. In view of this, this paper presents an investigative analysis of using Light Fidelity (Li-Fi) as an alternative to Wireless Fidelity (Wi-Fi) in 2-dimensional and 3-dimensional image transmission. A Li-Fi transmitter and receiver circuit diagram were developed and implemented for the purpose of the transmission. A program interface for the Li-Fi circuitry was implemented using MATLAB 2016b software. The conversion of the input image was achieved using the image processing toolbox in the MATLAB software. Therefore, it can be concluded that the channel provides high security due to the fact that light rays do not penetrate walls as well as the reasonable bandwidth required to carry both compressed and uncompressed images (2D/3D). Thus, making it suitable for image transmission.

Keywords: Li-Fi, Wi-Fi, 2D, and 3D image transmission

Introduction

Cisco which is one of the largest internet providers in recent times, forecasted in 2018 that the Global IP traffic will be 21 billion networked devices as against the 12 billion obtainable in 2013 (Nivrutti and Nimbalkar, 2013; Khandal and Jain, 2014). This means that the majority of networked devices will be mobile and connected through a wireless channel. As such, making the current radio frequency band overcrowded (Singh and Veeraiahgari, 2014). An alternative to reducing the over-crowding of the frequency band is with the use of Visible Light Communication (VLC) as presented by the authors in Dutta *et al.* (2014). The VLC technology is relatively a new concept popularly known as Light Fidelity (Li-Fi) coined by Harald Hass in 2013. The Li-Fi concept uses a transmitter which are the light bulbs or LEDs that modulates the signal to be transmitted over the channel. The receiver (a photon detector) is used to capture the sent signal and process the information (Gawande *et al.*, 2016). However, the transmitter and receiver can only communicate in a Line of Sight (LoS) condition.

The conceptual diagram that shows the functional and building blocks of the Li-Fi as a communication solution is depicted in Fig. 1 (Jadhav *et al.*, 2017).

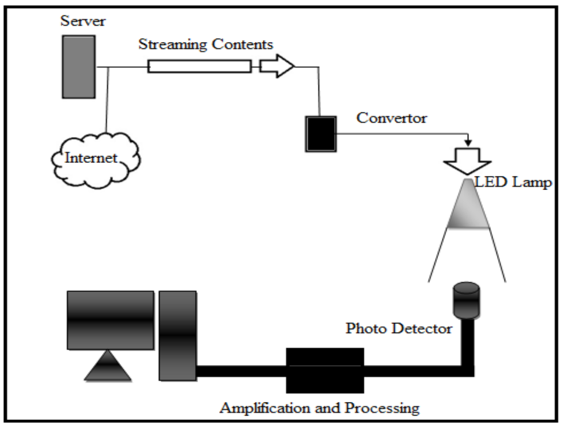

Fig. 1: Conceptual diagram of a Li-Fi system

Some research works already demonstrated the applicability of Li-Fi communication channel as an alternative to Wi-Fi, such as the works of Karunatilaka *et al.* (2015); Rashmi *et al.* (2015); Gawande *et al.* (2016); Khare *et al.* (2016); Vijay and Geetha (2016); Jadhav *et al.* (2017). These authors designed different transmission and receiver circuit diagrams with respect to their application domain. As such, from these works, it could be generalized that there is no universal Li-Fi transceiver that is suitable for all application domain as a connectivity paradigm. Therefore, researchers should design their own transceiver with respect to their application domain to achieve higher efficiency (Vijay and Geetha, 2016; Khan, 2017; Cailean and Dimian, 2017)).

This paper presents a transceiver circuit diagram for 2D/3D image transmission using the Li-Fi channel. The properties of an image transmission channel such as security, bandwidth (Mahesh *et al.*, 2016) among others, makes Li-Fi suitable for image transmission. Images were acquired with a camera module attached to MATLAB 2016b simulation software in 3D. Conversion of the image from 3D to 2D was done using the image processing toolbox in the MATLAB software before transmitted over the Li-Fi channel.

The remaining aspect of the paper is organized as follows: section two presents the materials and methods used to set up the developed transceiver circuit in the laboratory. Section three shows the result and discussion drawn from the experimental setup. Section four conclude the investigative analysis of using the Li-Fi to transmit 2D/3D images.

Materials and Methods

The following materials were used in this work:

a) A computer system with MATLAB software installed and connected to the developed Li-Fi transceiver.
b) A camera module used to capture images
c) LEDs and photon detectors which are the key components that form the developed circuitry.

The diagram of the developed transmitter and receiver circuit diagram are presented in Figs. 2 and 3.

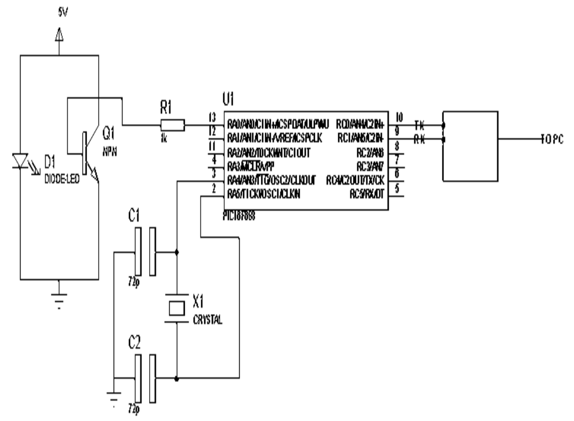

Fig. 2: Li-Fi transmitter circuit

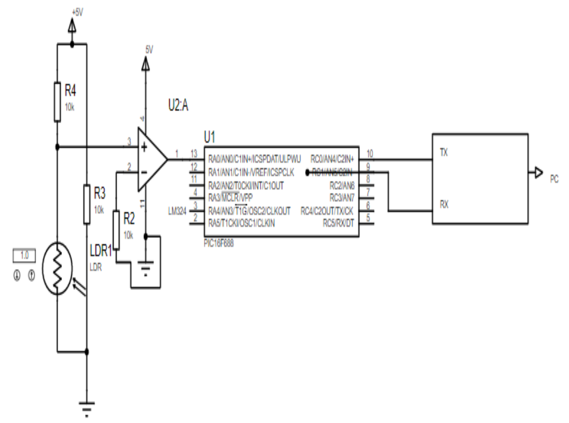

Fig. 3: Li-Fi receiver circuit

The list of electronic components used in the development of the transmitter section is listed as follows:

1. Microcontroller (PIC16F688)
2. USB to Serial Converter (MAX 232)
3. Resistors (1k)
4. LED
5. NPN Transistor
6. Crystal oscillator
7. Capacitors (72pf)

The list of electronic components used in the development of the receiver section are listed as follows:

1. Microcontroller (PIC16F688)
2. USB to Serial Converter (MAX 232)
3. The comparator (LM324)
4. Variable resistor (10k)
5. Phototransistor

In the operating principle, a transmitter code developed using the MATLAB software and as presented in the Appendix will execute in the OS and will take the image as an input by splitting it into pixels and then transmitting it through the serial port. The image was first transmitted as a 3D and then a 2D.
The image data are encoded serially from the computer using the LED and then received by the photon detector.
A signal converter decodes the sent data as RS 232 signal. A receiver code program at the other end to convert this serially received pixel data back into an image and then display it on the computer.

Result and Discussion

The realization of the design, as well as detail implementation, is discussed subsequently.
Temporary connections were established using a breadboard in the laboratory. Firstly, the microcontroller socket was placed on the breadboard taking note of the continuity in the breadboard. The IC or Microcontroller is placed on the socket. The USB serial converter was connected to the pin 10 and 9 of the microcontroller. The USB to serial converter has a path that will link up the PC to the transmitter circuit. Also, an NPN transistor base terminal is connected to pin 13 of the microcontroller through a resistor of 1 k value, the source is connected to a five-volt supply while the drain is grounded. The LED responsible for the final output is connected across the transistor. This set up makes up the section responsible for transmitting the data as depicted in Fig. 4.

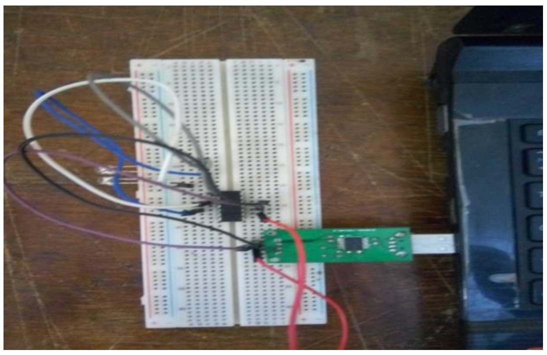

Fig. 4: Completed construction of the transmitter

On the receiving side, Pin 9 and 10 of the microcontroller was connected to the USB to serial converter which provides a link to the PC. The comparator output terminal is connected to pin 13 of the microcontroller while the positive input terminal is connected to the source terminal of the phototransistor which is parallel with the combination of a series dual 10k resistors which are in tune connected with the negative terminal of the comparator and then to ground. The VCC terminal of the comparator is connected to pin 1 of the microcontroller that generates a 5V power supply as well as the source terminal of the phototransistor. Fig. 5 presents the completed construction of the receiver.

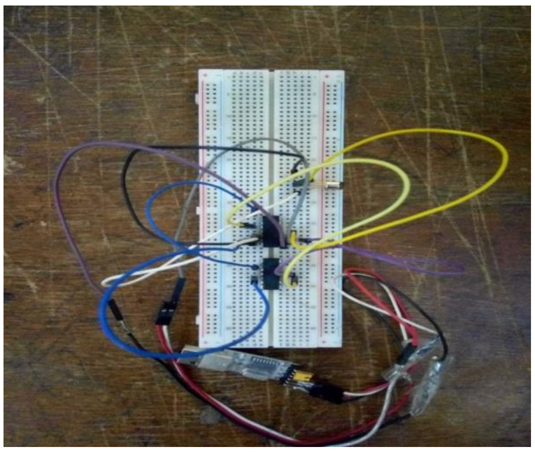

Fig. 5: Completed construction of the receiver

The selected image is loaded in MATLAB & initialization of the serial port is done using the MATLAB code in Appendix. LEDs act as an optical transmitter while the phototransistor (L14G2) acts as an optical receiver and sends the data to the microcontroller which again converts this binary data back to decimal form. Data is serially received using Visual Basic (VB) and MATLAB receives this data through VB text box linked to it. MATLAB then reorders the pixel data into an image. The output image is displayed after image reconstruction. Figs. 6 and 7 present the 3D and 2D image transmission scheme.

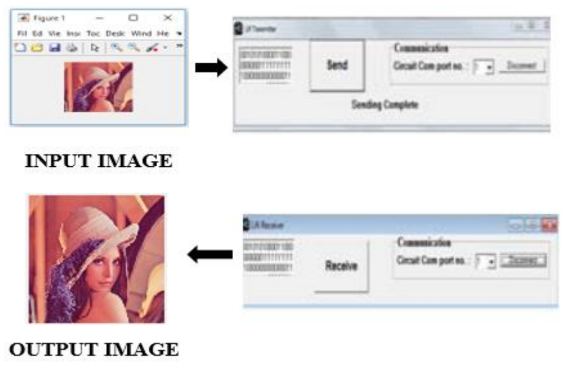

Fig. 6: 3D image transmission

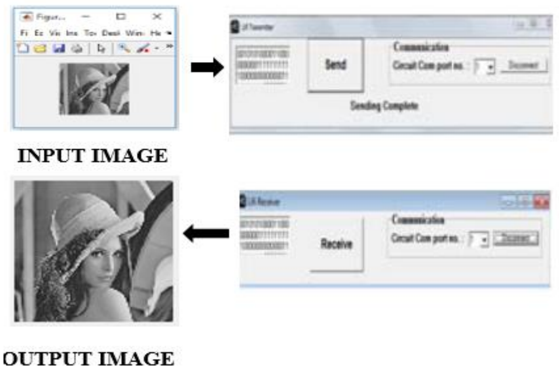

Fig. 7: 2D image transmission

The output result showed that the Li-Fi channel satisfied the requirement of image transmission as presented in (Mahesh *et al.*, 2016). This is because the images in 2D and 3D that were sent by the transmitter were successfully received provided there is a line-of-sight between them. Also, to ensure better communication between the transmitter and receiver, there is a need to channel the light source. Therefore, based on this investigative analysis, Li-Fi can serve as a substitute to Wi-Fi in the transmission of images to reduce the overcrowding of the frequency band in Wi-Fi.

Conclusion

In conclusion, from the implemented Li-Fi transmitter and receiver circuit diagram image transmission is largely achievable. The channel provides high security due to the fact that light rays do not penetrate walls as well as the reasonable bandwidth required to carry both compressed and uncompressed images (2D/3D). Further work can consider designing a Li-Fi circuitry for full duplex transmission.

References

Cailean AM & Dimian M 2017. Impact of IEEE 802.15.7 Standard on Visible Light Communications Usage in Automotive Applications. *IEEE Communications Magazine*. Retrieved from http://scholar.google.com, December 2017.

Dutta S, Sharma K, Gupta N & Bodh TL 2013. Li-Fi (Light Fidelity)- A new paradigm in wireless communication. *Int. J. Innovative Res. Computer and Commun. Engr.*, 1(1): 1654-1658.

Gawande P, Sharma A & Kushwaha P 2016. Various modulation techniques for LiFi. *Int. J. Advanced Res. Computer and Commun. Engr.*, 5(3): 121-125.

Jadhav P, Khatib S & Maner K 2017. DATA Transmission Through Li-Fi. *Int. Res. J. Engr. and Techn. (IRJET)*, 4(2): 1751-1754.

Karunatilaka D, Zafar F, Kalavally V & Parthiban R 2015. LED-based indoor visible light communications: State of the art. *IEEE Commun. Surveys & Tutorials*, 17(3): 1649-1677.

Khare Y, Tiwari VP, Patil AB & Bala K 2016. Li-Fi technology, implementations, and applications. *Int. Res. J. Engr. and Techn. (IRJET)*, 3(4): 1391-1394

Khandal D & Jain S 2014. Li-Fi (Light Fidelity): The future technology in wireless communication. *Int. J. Information & Computing Techn.*, 4(16): 1688-1693.

Khan LU 2017. Visible light communication: applications, architecture, standardization, and research challenges. *Elsevier J. Digital Commun. and Networks*, 3(1): 78-88.

Mahesh C, Agrawal D & Atul B 2016. Image transmission through wireless channel: A review. *IEEE Conference on Power Electronics, Intelligent Control, and Energy Systems*. Retrieved from https://ieeexplore.ieee.org/document/7853121, December 2017.

Nivrutti DV, Nimbalkar RR 2013. Light-Fidelity: A reconnaissance of future technology. *Int. J. Advanced Res. Computer Sci. and Software Engr. (IJARCSSE)*, 3(11): 753-758.

Rashmi R & Balaji B 2015. Prototype model of Li-Fi technology using visible light communication. *Int. J. Electrical, Computing Engr, and Commun.*, 1(4): 1-5.

Schmid S & Corbellini G 2013. LED-to-LED visible light communication networks. *ACM Transactions on communication Networks*, 1(1): 1-9.

Singh AJ & Veeraiahgari B 2014. Light fidelity technology. *IOSR J. Electronics and Commun. Engr. (IOSR-JECE)*, 9(2): 115-118.

Vijay, R & Geetha S 2016. Green environment monitoring system (Gems) for industries using Li-Fi technology. *Indian J. Sci. and Techn.*, 9(48): 1-6.

APPENDIX

Source Code

MATLAB code for the transmitting side;

```
functionvarargout = gui_tx(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @gui_tx_OpeningFcn, ...
'gui_OutputFcn',  @gui_tx_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
```

```
'gui_Callback',   []);
ifnargin&&ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
ifnargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
functiongui_tx_OpeningFcn(hObject, eventdata, handles,
varargin)
movegui(hObject,'center')
set(handles.tx_b,'Enable','off')
% Choose default command line output for gui_tx
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% --- Outputs from this function are returned to the command
line.
functionvarargout = gui_tx_OutputFcn(hObject, eventdata,
handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
functioncom_ser_Callback(hObject, eventdata, handles)
set(hObject,'Enable','off');pause(0.1)
com=get(handles.com_e,'String');
ifisempty(com)||isnan(str2double(com))
return
end
contents = get(handles.los_baud,'String');
vel=str2double(contents{get(handles.los_baud,'Value')});
handles.SerPIC = serial(['COM',com
set(handles.SerPIC,'BaudRate',vel);
set(handles.SerPIC,'DataBits',8);
set(handles.SerPIC,'Parity','none');
set(handles.SerPIC,'StopBits',1);
set(handles.SerPIC,'FlowControl','none');
fopen(handles.SerPIC);
warndlg(['Port COM',com,' OPEN'],'RS_232')
set(handles.tx_b,'Enable','on')
set(handles.los_baud,'Enable','off')
guidata(hObject,handles)
function figure1_CloseRequestFcn(hObject, eventdata,
handles)
try
fclose(handles.SerPIC);
catch
end
delete(hObject);
functiontx_b_Callback(hObject, eventdata, handles)
mensaje=get(handles.editor,'String');
long=length(mensaje);
for n=1:long
fprintf(handles.SerPIC,'%s',mensaje(n))
end
fwrite(handles.SerPIC,13)
set(handles.editor,'String',[]);
functionclc_b_Callback(hObject, eventdata, handles)
set(handles.editor,'String','');
functioncerrar_b_Callback(hObject, eventdata, handles)
ifstrcmp(handles.SerPIC.Status,'open')
fclose(handles.SerPIC)
set(handles.com_ser,'Enable','on')    warndlg(['Port COM','
CLOSE'],'RS_232')
end
set(handles.los_baud,'Enable','on')
set(handles.tx_b,'Enable','off')
```

Matlab code for the receiver side

At the receiving end the following code is used;

```
functionvarargout = gui_rx(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @gui_rx_OpeningFcn, ...
'gui_OutputFcn',  @gui_rx_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
ifnargin&&ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
ifnargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
functiongui_rx_OpeningFcn(hObject, eventdata, handles,
varargin)
movegui(hObject,'center')
set(handles.rx_b,'Enable','off')
set(handles.stop_b,'Enable','off')
handles.acc=[ ];
% Choose default command line output for gui_rx
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% --- Outputs from this function are returned to the command
line.
functionvarargout = gui_rx_OutputFcn(hObject, eventdata,
handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
functioncom_ser_Callback(hObject, eventdata, handles)
set(hObject,'Enable','off')
pause(0.1)
com=get(handles.com_e,'String');
ifisempty(com)||isnan(str2double(com))
return
end
contents = get(handles.los_baud,'String');
vel=str2double(contents{get(handles.los_baud,'Value')});
handles.SerPIC = serial(['COM',com]);
set(handles.SerPIC,'BaudRate',vel);
set(handles.SerPIC,'DataBits',8);
set(handles.SerPIC,'Parity','none');
set(handles.SerPIC,'StopBits',1);
set(handles.SerPIC,'FlowControl','none');
set(handles.SerPIC,'Terminator',13);
set(handles.SerPIC,'TimeOut',2);
fopen(handles.SerPIC);
msgbox(['Port COM',com,' Enabled'],'RS_232')
set(handles.rx_b,'Enable','oN')
set(handles.stop_b,'Enable','on')
guidata(hObject,handles)
function figure1_CloseRequestFcn(hObject, eventdata,
handles)
try
fclose(handles.SerPIC);catch
end
delete(hObject);
functionrx_b_Callback(hObject, eventdata, handles)
set(handles.editor,'String','')
set(handles.rx_b,'Enable','off'); pause(0.1)
handles.acc=[ ];
set(handles.stop_b,'UserData',0)
while 1
if get(handles.stop_b,'UserData')
break
end
ifhandles.SerPIC.BytesAvailable
    A=fscanf(handles.SerPIC);
handles.acc=[handles.acc, A];
set(handles.editor,'String',handles.acc)
end
pause(0.02);
end
functionstop_b_Callback(hObject, eventdata, handles)
set(handles.stop_b,'UserData',1)
ifstrcmp(handles.SerPIC.Status,'open')
fclose(handles.SerPIC);
end
set(handles.rx_b,'Enable','off')
set(handles.com_ser,'Enable','on')
set(handles.editor,'String','')
warndlg('Reception END','WARNING RS-232')
guidata(hObject,handles)
```