Using Mobile Phone Based Camera to Read Information from a Li-Fi Source

Using Mobile Phone Based Camera To Read Information From A Li-Fi Source

Sreesha Damodaran
School of Mathematical and Computer Sciences
Heriot Watt University
Dubai, U.A.E.
sreesha.damodaran@gmail.com

Talal Shaikh
School of Mathematical and Computer Sciences
Heriot Watt University
Dubai, U.A.E.
T.A.G.Shaikh@hw.ac.uk

Prof. Nicholas K Taylor
School of Mathematical and Computer Sciences
Heriot Watt University
Dubai, U.A.E.
n.k.taylor@hw.ac.uk

***Abstract* - At crowded events, an increasing number of people use their mobile devices at the same time to run applications having high data requirements leading to network congestion. This paper investigates the use of LED light to act as a Li-Fi source in order to transmit data. The data is captured using the CMOS camera of an Android smartphone. The experimental setup is tested using two different Android devices and the results are compared.**

Keywords—Li-Fi; LED; Network Congestion; Arduino Transmitter; Safe Transmission of Data; Mobile Phone; Android

I. INTRODUCTION

The advent of handheld cellphone device with "DynaTAC" (DYNamic Adaptive Total Area Coverage) by Motorola in the 1970s initiated the start of commercial mobile communications. Over the past forty years, wireless connectivity has turned into a necessity much like the fundamental utilities.

With the number of users using smartphones and tablets increasing exponentially every year, users are employing web services through various applications having very high data requirements. There are over 1.4 million cellular radio base stations all around the world, with an estimate of 5.5 billion mobile users by 2020.

With mobile devices transmitting over 600 TB of data - spectrum is essential as wireless communication requires the use of radio waves. This leads to the event handling organizations compete for wider network coverage leading to failure of network services due to overloading of the mobile bandwidth, leading to a shortage in Radio Frequency Spectrum.

II. RADIO SPECTRUM CONGESTION

Impact of visual networking applications on global networks is tracked and gauged by Cisco Visual Networking Index (VNI) Global Mobile Data Traffic Forecast Update (part of Cisco VNI Forecast). Cisco VNI forecasted that by 2020, global data traffic will cross 30.6 Exabytes per month at a compound annual growth rate (CAGR) of 53% from 2015 to 2020 (1 Exabytes = 1018 bytes). This in itself is an eightfold increase from 2015. The estimated increase in mobile data usage (data rate in Exabytes) over a span of five years ranging from 2015 to 2020 is shown in Fig 1 [1].

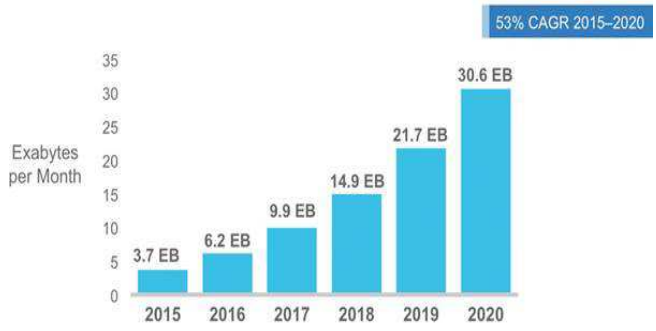

Fig 1: Cisco VNI Prediction of data usage from 2015- 2020 [1]

Cisco VNI also predicted that by 2020, even though the 4G connections will account for 40.5% of the connections, 4G mobile services will account for 72% of total traffic. That is, by 2020, on average 4G connection will generate 3.3 times more than data traffic than non 4G connection. Another key point to note was at the rate of 71% CGAR, Middle East and African regions have the strongest mobile data traffic growth compared to other regions (Refer Fig 2). Asia Pacific and Central and Eastern Europe will follow at 54% and 52% CGAR respectively [1].

Fig 2: Global Mobile Data Traffic Forecast by Region [1]

With the number of mobile devices increasing ever exponentially with high requirements for data related services and applications, the congestion of radio spectrum poses a real problem. An emerging technology called Light Fidelity, or better known as Li-Fi, was demonstrated by Dr. Harald Haas in 2011 TED Talk. This technology aims to use Visible Light Spectrum instead of Radio Spectrum.

III. LIGHT FIDELITY TECHNOLOGY

Dr. Harald Haas in his TED Global Talk (2011) on Visible Light Communications proposed a technology in order to overcome the radio spectrum congestion. In this talk, Dr. Haas demonstrated his invention called D-Light, which he also referred to as "data through illumination", by transferring data at speeds exceeding 10 Mbps using light waves from an ordinary table lamp to a nearby computer. Light Fidelity or Li-Fi stands for the data transmission through visible light and is therefore considered as an optical version of Wireless Fidelity or Wi-Fi [2][3].

Li-Fi is usually implemented using LED lights, which can be white, at the transmitter side. Normally, LED light bulbs are used as a source of illumination by applying a constant current. However, by varying the current too fast, LED light bulb's output can be made to fluctuate at extremely high speeds. If the LED is switched on then the data transmitted is a digital one and if the LED is switched off, then the data transmitted is zero.

Further variations can be made to this setup by using an array of LEDs for parallel data transmission, or using a combination of red, green and blue LEDs and altering each light's frequency with each of these frequency encoding a different data channel. Such variations can promise transfer of data at a speed of 10Gbps or more.

Since the carrier in Li-Fi technology is visible light, Li-Fi sources are intolerant to noises and disturbances thereby increasing security – as light cannot penetrate walls. The use of LED light bulbs could mean that every street lamp could be a potential access point [2].

Even though there are many advantages of Li-Fi technology, the inability of light to penetrate walls and other opaque materials is considered as one of the technology's disadvantages. Another disadvantage would be that Li-Fi only works in Line of Sight (LOS) [3].

However, comparing Wi-Fi and Li-Fi technologies, Li-Fi technology aims to resolve many issues faced by Wi-Fi technology mainly in terms of capacity, efficiency, availability and security. With over 10,000 times more bandwidth than Radio Spectrum, there is no concern over the bandwidth overcrowding. Also, data transmission using LED light bulbs would be less expensive but highly efficient and can provide illumination as well as transfer data.

A. Principle of Light Fidelity

The key component of LiFi technology is Light Emitting Diodes (LED) light bulbs. Data is encoded in light by operation LED light bulbs at very high speeds using data encoding techniques. The main advantage of using LED light bulbs is that it can produce high brightness and hence provide illumination. Since the operating speeds of LEDs are less than 1 µs, on and off keying at very high speeds undetectable to the human eye is possible [4][5].

A controller is needed to transmit data along with LED light bulb. Depending on the data to be transmitted, the rate of switching on and off the LED light bulb can be varied. Usually an array of LED light bulbs is employed to increase the rate of data transmission. The different colored LEDs in the array can be attributed to different data channels [6].

Data, source of which can either be internet or a web server can be encoded in LED light bulb and transmitted. Data encoding can be achieved at the transmitting end by flickering the LED light bulbs at very high speeds invisible to the human eye or by varying the light bulb intensity. This light with the encoded data in it is then received at the receiving end by a photo detector. The photo detector detects and decodes the data, amplifies it and sends it to the end receiver which can either be a mobile device or a computer [7].

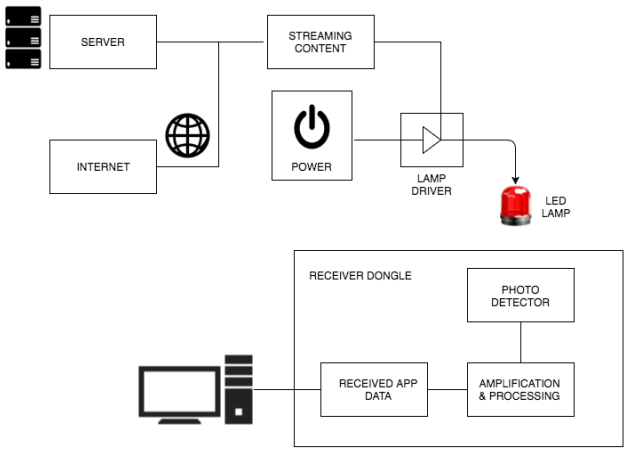

Fig 3: Working Scenario of Li-Fi technology

The lamp driver contains the streamed and encoded data from the internet. The LED light bulb used to transmit the data contains a microchip. This microchip on switching the LED light converts the digital data to light waves which are then transmitted. These transmitted waves are received at the receiving end by a photodetector, which is then amplified, decoded and converted back to its original format. This message is then transmitted to the end receiver which can either be a mobile device or a computer. Working architecture of data transmission using LED light source is explained in Fig 3 above [7].

IV. RELATED WORKS

Ong and Chung have developed a long range VLC temperature monitoring system employing the CMOS camera sensor in an Android mobile device. The Android device used for this development is Samsung Galaxy Note 10.1. Three different colored LEDs were used in this experiment for transmitting data. Color detection at the

receiver end was done by calculating the reference boundary and using Gaussian Blur technique in order to reduce noise and detail. This technique only transferred two character digits [8].

Another way of using CMOS camera sensor of the mobile device is to convert the image captured by the camera into a grayscale image which consists of YUV components. The Y component is referred to as Luminance and determines the brightness of the color while the U and V component determine the Chroma or the color itself. From the grayscale image, the Y component is extracted using image processing techniques. The dark bands in the image will represent the data bit when it zero and the light bands will represent the data bit one. This technique was used by Danakis et al. in his paper [9].

V. Proposed System Architecture

The proposed system architecture consists of a transmitter and receiver setup. The transmitter end consists of an Arduino circuit with a basic LED blinker circuit. The receiver end consists of an Android mobile application running in the end user's mobile device. The application is responsible for receiving the data bits from the transmitter end, decoding the bits on the fly and then displaying the result on the mobile device screen.

The LED light in the Arduino setup encodes the data in it using encoding techniques. Fig 4 below shows a scenario where the above proposed system architecture can be deployed. The scenario under consideration here is an exhibition where the crowds are usually large in numbers.

The network congestion in crowded places is higher as all the users are using their mobile devices at the same time. The proposed system aims to relieve the network congestion and provide secure data transmission to the end users. LED acts as Li-Fi source in this case. The LED light can also provide illumination thereby improving power efficiency.

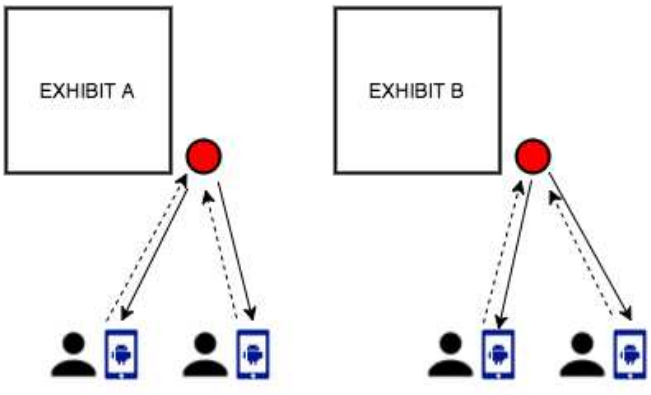

Fig 4: Proposed Working Architecture

Once a user comes in front of the exhibit and launches the Android mobile application from his/her phone, the application starts and the camera preview comes. The application then recognizes the LED data light place next to the exhibit, decodes the data which is then displayed. This data can be additional information and history regarding the exhibit under consideration. One Li-Fi source can transmit data to five different mobile devices at the same time.

The users can also interact and stream data using this application. The proposed system consists of a transmitter and receiver which is explained in detail in the next section.

A. Data Encoding

The transmitter end of the proposed system consists of an Arduino microcontroller used to the control the LED light. The data to be transmitted is encoded in the fluorescent light or LED light. There are two ways of encoding data in LED light: one way is by flickering the LED at very high speeds that are unperceivable to human eye, the second way to encode data is through frequency variation. Modifications are made to the amplitude and frequency of the arc current by controlling the arc frequency of the lamp. On reaching the appropriate value, the LED will light up [10].

The usual approach to creating an Arduino transmitter setup for Li-fi data transmission requires the use of a lamp driver and a microchip in the LED light, but in this case the approach is to create a blinker circuit in Arduino. This circuit will light the LED on when the data to be transmitted is one and otherwise zero. The data to be transmitted to the receiver end can either be hardcoded in the program or can be input through the Serial Monitor of the Arduino Microcontroller. This simple method is used in this project to the test the system out. The circuit diagram for the same is shown below (Fig 5).

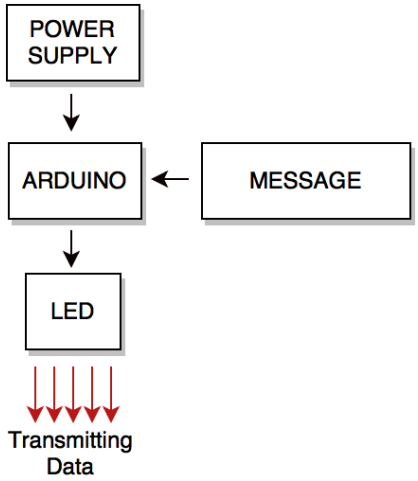

Fig 5: Transmitter Circuit

The data considered for testing this experimental setup initially is only text based data. The circuit setup is similar to the blinker circuit in Arduino and consists of one single LED transmitting data from an external pin. There are different ways of transmitting the encoded data. The data could be hardcoded with the LED transmitting the same text every time or a serial monitor could be used to send the text.

The header of the text message contains header bits of four ones each ("1111") and two trailer bits ("10"). These header and trailer bits along with the text data in binary format is transmitted. A total of 30 bits are transmitted by the

LED light continuously with 6 bits being used for verification and 24 bits for data.

A delay of 143 milliseconds is used at the transmitting end in order to match an average of 7 frames per second at the receiving end mobile device. This is to accommodate the slower processing rate at the Receiver end.

B. Receiving End

The receiving end of the application consists of an Android mobile application which receives and decodes the text message transmitted by the transmitter circuit setup.

In order to successfully receive the data bits and decode the text message, color blob detection method was used. In this method, the red color of the LED light bulb was used for detecting the data bits.

The external light settings of the environmental setup was set constant. The application detected the red color blob of the LED light bulb. Then contours were drawn around the blob and counted. If the contour size is greater than one, then the LED is lit and if the contour size is equal to zero then the LED is considered to be zero.

This approach was preferred since this proved to detect the state of the LED accurately and the frame rates were considerably higher with an average (being around 6 to 7 frames per second).

Once the state of the LED was detected, the next step was to choose an appropriate data type in order to store these values. The main challenge to implement this functionality was that since the input frame are live frames, storing these frames and then processing them proved to be quite difficult. After storing the states of the data frames, the frames were further send for decoding on the receiver end. These operations are happening in the main thread of the application and can considerably increase the processing speed.

VI. METHODOLOGY

Compared to other works, this paper differs in the way the color detection is done at the receiving end of the experiment. Different techniques of color detection were tried and tested. OpenCV Image Processing Library is used, wherein instead of using the traditional camera of the Android device, JavaCV camera view of the OpenCV is used.

The JavaCV camera view then detects the red color of the LED light. The next step was to draw contour around the red LED light when the LED is on. The contours are then counted and if the contour size is greater than one, then the data received is one and if not, then data received is zero. These values are then stored into an array and decoded to the text message.

VII. EXPERIMENTAL SETUP

The setup was tested on two Android Mobile devices with different specifications - Sony Xperia Z1 and Samsung S4. The setup was tested indoors under normal lighting conditions in different modes – fixed and hand held, in order to understand its effects on the overall setup.

Out of the two Android devices, Sony Xperia Z1 was kept fixed while the application was receiving and decoding the bits, while Samsung S4 was hand held during the data bits decoding process. In both the devices, the camera settings – White Balance and Exposure – were changed and the effects on the decoding end was recorded.

A. Comparing Z1 and S4

TABLE I. COMPARISON BETWEEN Z1 AND S4

Comparison		
	Galaxy S4	***Sony Xperia Z1***
Sensor	13MP 1/3.06" (4.69 x 3.52 mm)	20.7MP 1/2.3" Exmor RS (6.17x4.55 mm)
Sensor technology	Back-illuminated (BSI)	Back-illuminated (BSI)
Pixel size / pixel pitch in microns	1.12 µ	1.175 µ
Lens focal length	4.2mm (31mm equivalent) Largan Precision Optics	27mm equivalent Sony G lens (5 elements)
Maximum lens aperture	f/2.2	f/2.0
Image stabilization	No optical image stabilization (IBIS) Digital image stabilization for videos	No optical image stabilization (IBIS) Digital image stabilization for videos
Flash	Single LED Flash	Single LED flash
Continuous shooting (burst)	20 fps in regular burst mode (full-res images at 13 megapixels)	30.5 fps using Timeshift
Display	5 inches 1080x1920 pixels 441 ppi Super AMOLED Gorilla Glass 3	5 inches 1080x1920 pixels 441 ppi Sony TRILUMINOS display (TFT)

Sensor size along with the size of each photodiode (light sensitive pixel) are important factors when it comes to producing high quality images. Xperia Z1 features higher megapixel resolution, due to which there is much higher pixel density, therefore it has the smallest pixels resulting in producing high quality images.

Both the devices lack in the true optical image stabilization mechanism. An optical image stabilization helps the device capture much better low-light photos due to better stabilization in low-shutter speeds. The lack of which therefore requires faster shutter speed limiting the amount of light that can reach the device camera sensor. Due to this particular reason, the setup was not tested in low light conditions.

VIII. RESULTS

In the case of Xperia Z1, the device was fixed when the data bits were being received from the transmitter end and decoded at the receiver end. White Balance and Exposure settings of the camera were changed and the setup was tested under normal lighting conditions. The following observations were made:

1. White Balance Set to Auto

The average frames per second (fps) dropped from the usual 30 fps to 7 fps. This decrease was attributed due to the fact that OpenCV Java Camera API was used. Since the processing occurred in real time, this considerably slowed down the fps. The error rate in decoding the bits from the actual input value was found to be 40%. The high error rate in the output can be due to the frame rate not being constant during processing and also due to the lack of synchronization between measuring the transmitter and the receiver.

2. White Balance Set to Cloudy Daylight

The average fps rate dropped down to 2 fps, with the OpenCV Java Camera API missing out on many frames from the transmitter end. The fps further dropped during the live processing on the receiver end on the fly. The Arduino setup is constantly transmitting the data bits with 143 milliseconds delay. As a result of which, the error rate was found to be 40% with increase in processing time.

3. White Balance Set to Daylight

In this scenario also, the frame rate dropped down to 2 fps during the processing time on the receiver end. As a result of which, many frames from the transmitter end was skipped by the camera on the receiving end resulting in error rate of 53.3% from the actual input value. This error can be attributed to decrease in the frame rate and lack of synchronization between the input and the output.

4. White Balance Set to Fluorescent

In this case, the frame rate dropped further down to 1 fps during the live processing on the receiver end but the error rate was only 20%. This decrease in error rate can be attributed to the frame rate staying constant during the processing time. Also, a level of synchronization was achieved between the transmitter and the receiver.

5. White Balance Set to Incandescent

The frame rate dropped from the usual 29 fps to an average value of 2 fps during live processing at the receiver end. But the error rate was close to 40%. The reason for the increase in error rate being that many frames are lost on the receiver end due to the processing being slower as it is running on the main thread of the program. Also, the lack of synchronization between the transmitter and receiver resulted in loss of many frames at the receiver end.

6. Exposure Set to Fluorescent

Even though the average fps during live processing was 5 fps, the corresponding error rate when decoding the data bits was found to be 50%. Since the average fps was comparatively better than in the other scenarios, the increase in error rate can only be attributed to the lack of synchronization between the transmitter circuit and the receiver setup.

7. White Balance and Exposure Set to Auto

In the last test case for fixed setup of Xperia Z1, both the exposure and white balance settings of the camera were changed and set to default Auto Mode in order to understand the effects of white balance and exposure on the receiver setup. The average fps during the processing phase dropped down to an average of 1 fps but the error rate increased to 46.6% on an average. This leads us to understand that setting the white balance and exposure to auto leads to no specific improvement in the test results achieved.

The next set of tests were performed using Samsung S4 in handheld setup during the processing phase. Since the device is handheld fluctuations in decoding the bits at the receiver end can be expected, along with decrease in the average fps as the processing happens in the main thread of the application.

In these test cases, only the white balance settings of the camera were changed for testing purposes.

1. White Balance Set to Auto

The average frame rate improved to an average of 6 fps compared to Xperia Z1. But the error rate during processing time increased to 53.3%. The increase can be attributed to the lack of synchronization between the transmitter circuit and the receiver setup decoding the data bits.

2. White Balance Set to Cloudy Daylight

The average frames per second remained at 6 fps and the error rate increased to 56.6% and can be attributed to firstly the setting of white balance to cloudy daylight leading to the environment being dark when capturing the live frames. This leads to the receiver end misreading few frames due to the bad light, thereby leading to increase in the error rate.

3. White Balance Set to Daylight

The average frames per second remained at 6 fps and the error rate considered decreased to 40%. The decrease in the error rate compared to cloudy daylight setting of white balance can be attributed to the environment setup. This setup offered better results compared to 53.3% than that we got in Sony Xperia Z1.

4. White Balance Set to Fluorescent

The average frames per second as in the previous scenarios remained constant at 6 fps which was considerably better than 1 fps in the case of Sony Xperia Z1. But the error rate reached 50% which is not reliable. The high error rate can be attributed to the lack of synchronization between the transmitter and the receiver end.

5. White Balance Set to Incandescent

Compared to the 40% error rate in the case of Sony Xperia Z1, the incandescent setting of white balance resulted in an error rate of 33.33%. The frames per second was better

averaging at 6 fps compared to 1 fps as in the case of Sony Xperia Z1.

IX. Observations

Setting the white balance was set to Auto in both the test devices, the error rate in Xperia Z1 was found to be 40% which is less compared to the 53.3% found in Samsung S4. This increase in the error rate can be attributed to lack of synchronization between the transmitter and the receiver end and also due to the fact that the device setup was different in both cases. The fluctuations when the device is handheld could have led to higher error rate in the case of Samsung S4.

Cloudy daylight white balance setting increased the error rate to 56.6% while the same settings in Sony Xperia Z1 was only 40%. Contrary to this result, daylight settings in Sony Xperia Z1 gave an error rate of 53.3% while Samsung, the error rate was only 40%. The high error rate could be due to the fluctuations when holding the device.

Comparing the test results from both the devices, incandescent setting in Sony Xperia Z1 gave an error rate of 40% while in the case of Samsung S4 device, the error rate was only 33.33%. This result was better compared to the other test case settings and can be attributed to the white balance settings of the device in itself playing a factor. Also, a level of synchronization was achieved between the transmitter and the receiver end of the setup resulting in decrease in error rate. The average frames per second was found to be better in the case of Samsung S4 with the fps averaging to 6 fps compared to 1 fps of Xperia Z1. This was favored as in the real world scenarios; the user would be holding the device rather than the setup being fixed.

X. Future Works

Future works on this project includes stabilizing the transmitter end of the circuit and reducing the processing time at the receiver end. The main thread of the Android application could be freed in order to decrease the processing time. This can be achieved by using the Android Native Development Kit specific to Android platform. Also, at the receiver end of the application, the use of error detection and correction techniques can be used in order to reduce the high error rate. A feedback loop can be used when the Android device is processing the data bits transmitted from the LED light at the transmitter end. The feedback loop will use error detection techniques in order to identify the error bit and also rectify it. The rectified data bits can then be decoded into the text data. Another technique that can be considered is synchronizing the transmitter and the camera of Android phone. This could prove extremely difficult in practical situations and also considering that every Android camera can have different specifications and different camera shutter rates. Other considerations for this experimental setup include the addition of multiple LEDs at the transmitter end, with each LED acting as different data link for transmitting data. The usage of multiple LEDs as in the work of Ong and Chung can be used for differentiating data at the receiving end of the application.

XI. Conclusion

We have explored the use of LED light bulbs for visible light communication with the receiver end being an CMOS camera of Android mobile device. The error rates incurred in data transfer were unacceptably high and we have investigated ways of mitigating this. There are further improvements that can be made in the experimental setup with the primary focus being on achieving synchronization between the Arduino setup and the mobile device in order to prevent loss of the camera frames. Also the main thread of the application can be freed to speed up the processing time. In addition to this, instead of trying to detect the red color blob of the LED, better techniques for detecting the LED light bulbs can be considered.

References

[1] CISCO, (2016). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014– 2019 White Paper.

[2] Sharma, V. (2014). Exploring Visible Spectrum:A Study Of Light Fidelity System and Its Applications. International Journal of Enhanced Research in Science Technology and Engineering, 3(5), pp.349-350.

[3] Tsonov, D., Videv, S. and Haas, H. (2013). Light Fidelity (Li-Fi): Towards All- Optical Networking. Institute for Digital Communications.

[4] Mutthamma, M. (2013) "A survey on Transmission of data through illumination - LiFi." IJRCCT [Online], 2.12 (2013): 1427-1430.

[5] Khandal, D. and Jain, S. (2014). Li-Fi(Light Fidelity): The Future Technology in Wireless Communication. International Journal Of Information and Computation Technology, 4(16), pp.192-193.

[6] Ubhale, P.R., Borkar, N.R., Baitule, M.A. (2014) Convergence of Wireless Communication : Light Fidelity (Li-Fi Technology) - Accessing Internet Via LED. International Journal for Engineering Applications and Technology.

[7] Pinglikar, A.R., Shandilya, P.A. (2014) LiFi (Light Fidelity) - Illuminating Civil Places for Communication. International Journal of Computer Engineering & Technology, [Online]. 5, 186-191

[8] Ong, Z., & Chung, W. (2016). Long Range VLC Temperature Monitoring System Using CMOS of Mobile Device Camera. *Sensors Journal, IEEE, 16*(6), 1508-1509.

[9] 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.

[10] Zalte, S. and Hatkar, A. (2015). Light Sensor Based Information Transmission System. IOSR - JECE, pp.1-4.