ISSN 1870-4069

Transmission and Reception of Images via Visible Light

Sergio Sandoval-Reyes

Instituto Politécnico Nacional, CIC, Mexico City, Mexico
sersand@cic.ipn.mx

Abstract. Communication by light or VLC by its acronym in English (Visible Light Communication), uses visible light from light emitting diodes (LEDs) to transmit information. Using a computing device and some hardware, the transmission of information is performed driving and modulating the light emitted by the LEDs. In the receiver side, the information carried by the modulated light is demodulated through a photo-detector, which is usually connected to a similar computing device for the final recovering of the information. In this article we describe an application based on VLC using OOK (On-Off Keying) modulation, to transmit color images from a RaspBerry Pi computer (using Python as the programming language), and several modules (LEDs and a sensor light) from LittleBits.

Keywords: VLC, image transmission, Raspberry Pi, Python, OOK.

1 Introduction

Visible Light Communication (VLC) [1, 2], can be used to transmit audio, voice and data. It uses laser light or light from emitter diodes (LEDs) and light detectors at the transmitter and receiver ends respectively (Fig. 1). It works in the 380 nm to 780 nm optical band which is visible light and hence the name VLC [3, 4, 5].

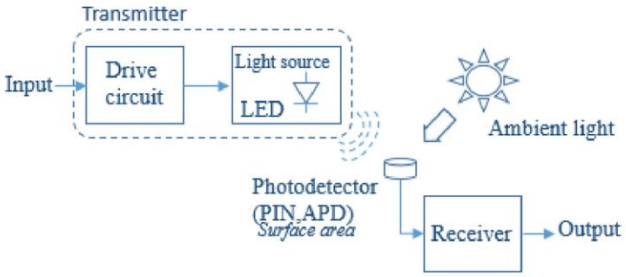

Fig. 1. A basic VLC link structure.

To convey information, this one has to be encoded, and then the light has to be modulated and demodulated at the transmitter and receiver sides. There are several methods to do this, some are briefly discussed in the following. Then, the received information has to be decoded and processed to recover it fully. The success of this

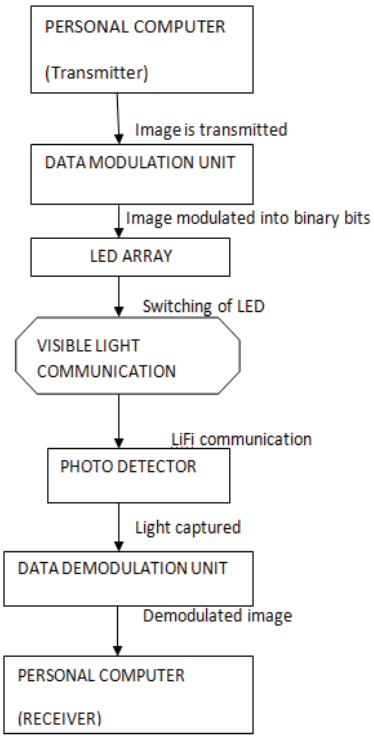

Fig. 2. A flow chart for VLC image transmission.

recovery depends of several factors, among them: 1) The number, shape, and wavelength of the LEDs employed; 2) The number and type of light detectors (photo-resistor, photo-diode, reverse-biased LED, etc.) used; 3) The encoding method (RZ, NRZ, NRZP, etc.); 4) The modulation scheme (OOK, WPM, VWPM, PPM, OFDM, etc.), and 5) The synchronization and distance between the LEDs and the light detector. Remember that VLC is a technology that among other things, it requires line-of-sight between emitter and receiver [6].

This paper describes an application based on VLC using OOK (On-Off Keying) modulation, to transmit as-a-proof of concept color images using a RaspBerry Pi 3 computer as the data source and sink (to simplify the synchronization problem between emitter-receiver), Python as the programming language, and several modules (LEDs and a sensor light) from LittleBits, to easy the hardware implementation.

The remainder of this paper is organized as follows: Section 2 gives a summary of works related to the transmission of digital images using VLC. Section 3 describes the design of our VLC application. Section 4 describes the experiments realized, the results obtained, and their analysis. Finally, our conclusions and future work are presented in Section 5.

2 Related Work

Several research works on VLC technologies to transmit images have been proposed. The most important are described in the following.

2.1 How Based VLC Image Systems Work

Typical based VLC image systems are implemented using an intensity modulation and direct detection (IM/DD) scheme with a line-of-sight (LOS) configuration [7]. In the transmitter, IM is implemented through the modulation of the transmitted signal into the instantaneous optical power of the LED by controlling the radiant intensity with the forward current through the LED (High modulation frequencies are used to avoid flicker). In the receiver, the transmitted signal is recovered using direct detection (DD). In this simple method, a photodiode (or array) is used to convert the incident optical signal power into a proportional current. Figure 2 shows a general flow chart for VLC image transmission [8].

2.2 VLC Image Transmitter

A typical based VLC image transmitter contains an image generator (a PC with Matlab to convert the image into bits), an interface (usually a USB cable) to send the bits to a microcontroller (for coding and modulation), and outputted through one of its ports to the LED driver and the LED optics. See Figure 3. [9]. The modulated signals are used to switch on-an-off the LEDs at desired frequencies using LED drivers. These drivers rely on trans-conductance amplifiers to convert voltage signals into corresponding current signals to excite the LEDs array for communication purposes.

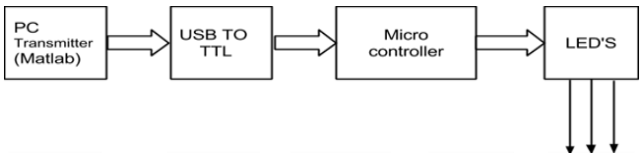

Fig. 3. VLC image transmission.

2.2.1 VLC Modulation

Although there are different modulation schemes for VLC, mainly, on-off keying (OOK), variable pulse-position modulation (VPPM), color shift keying (CSK) and orthogonal frequency division multiplexing (OFDM) [10], OOK is the most popular. OOK is the most commonly used IM/DD modulation scheme in VLC due to its simple implementation. In this method basically the LED intensity is changed between two distinguishable levels corresponding to the data bits (1 or 0). See Figure 4. A modified OOK, called Variable OOK (VOOK) can provide dimming. It is achieved by changing the data duty cycle through pulse-width modulation (PWM), with only 1 bit of information carried per symbol period.

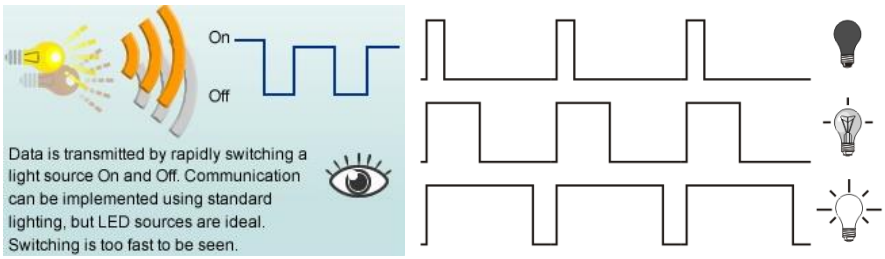

Fig. 4. On-Off Keying with PWM.

2.3 VLC Image Receiver

A typical optical image receiver consists of a photo detector followed by an amplifier. The photo detector can be a photo transistor, a reverse-biased LED, or a Light Detect Resistor (LDR). The light captured by the photo transistor which acts as a sensor, passes the output to the comparator which compares the binary input, and similarly the original image is recovered using Matlab software in the PC. See Figure 5 [8, 9].

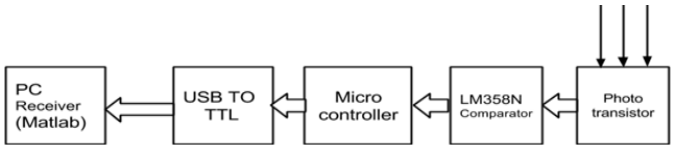

Fig. 5. VLC image reception.

3 Image Transmission and Recovering Using VLC

In the following we develop a VLC application to transmit and receive in real time an image using two LEDs in the transmitter, and a photo detector in the receiver.

In order to do that, we will use in the transmitter as a data source, a Raspberry Pi 3 (RBPi) computer, and five LittleBits bit-modules [12]: power, button, proto, split, and two bright LED bits. See Figure 6. The RBPi using a software written in Python will read an image byte by byte, and will send each byte using the SPI MOSI (Master Output Slave Input) output port (Pin 19), toward the proto module, OOK signals to drive the two LEDs.

While as in the receiver we will use three LittleBits components: power, light sensor, and another proto bit. See Figure 7. The light sensor captures the light emitted by the two LEDs and converts it into a digital signal which is fed to the proto module. The proto module in turn outputs this signal and with a wire connector, feeds this signal toward the MISO (Master Input Slave Output) input port (Pin 21) of the RBPi.

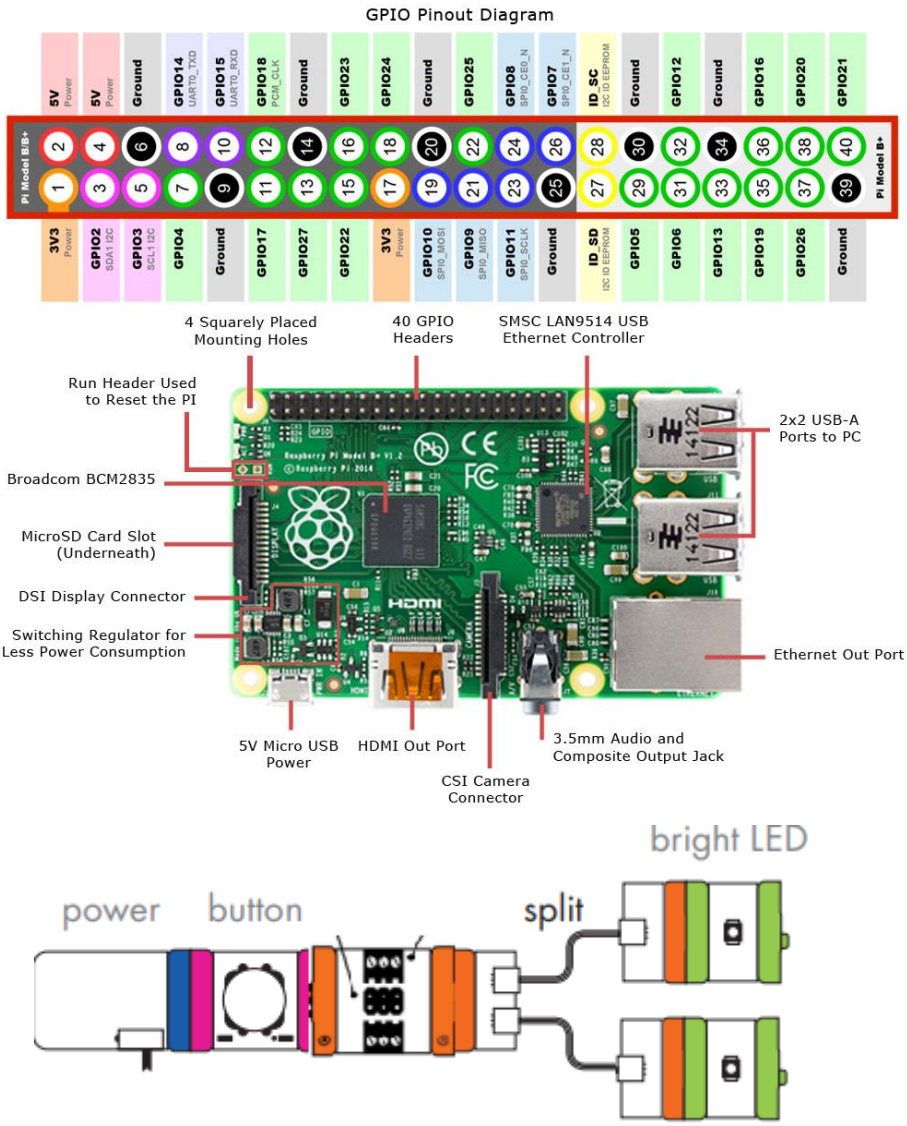

Fig. 6. Raspberry Pi 3, and five bits: power, button, proto, split and two LEDS.

Fig. 7. Receiver components: Light sensor and proto bits.

3.1 Transmitting an Image Via VLC

To transmit an image via VLC, the RBPi with a script written in Python, opens a picture file ("lenna.jpeg") and it reads it into a bytearray "b". Then with a "for" loop reads byte by byte of the picture and it send them to the SPI MOSI output port 19, using the spi.xfer () directive. This MOSI output is fed using two wire connectors (signal and ground), into the input of the "proto" module (lower middle connector in figure 7). The proto module outputs and split the OOK signals to drive the two LEDs. The Python code to execute the above mentioned is shown in Figure 8.

```
# Python code to read, transmit, and recover a picture via Visible Light
import spidev
from PIL import Image, ImageFilter
import io
from array import array
spi = spidev.SpiDev()
spi.open(0,0)
spi.max_speed_hz = 4000000
b = bytearray()
buffer = bytearray()
try:
    # Load an image from Raspberry Pi
    with open ("lenna.jpeg", "rb") as img:
        f = img.read()
        b = bytearray(f)
    # Sending with SPI and converting the image, byte by byte into visible light
    for byte in b:
        rx = spi.xfer([byte])
        rx_data = rx[0]
        buffer.append(rx_data)
    # Recovering the bytes array and back into an image
    img_rec = Image.open(io.BytesIO(buffer))
    img_rec.save(´lena_img_recovered.png´)
    img_rec.show()
except:
    print "Unable to recover image"
```

Fig. 8. Python code to read, transmit and recover a picture via VLC.

In this code, it is necessary to import the following libraries: "SPI", "PIL" (Python Image Library), and "Array". The use of the LittleBits modules simplified very much the hardware implementation. The power module was fed with a 9-V battery, and this was necessary because the outputs of the RBPi are low-voltage (3.3 volts) and low-current (Individual pins must not pull more than 16 mA and the entire GPIO must not source more than 50 mA), which are no good enough to drive two bright LEDS [13]. These LEDs are simple yellow LEDS with a wavelength of 550-to-600 nm, luminous flux of 4-to-5 lm, and consume around 16-to-20 mA each, with an aperture angle of about 120 degrees. See Figure 9.

Fig. 9. LittleBits bright LED.

3.2 Receiving an Image via VLC

As was mentioned, the picture was sent via VLC as LED light. This light is received through a light sensor module which then sent it back through out another proto module, to the MISO port 21 of the RBPi. This light sensor not only receives the OOK light signal but also has a trans-impedance amplifier for high speed operation. The light sensor has 2 modes, Figure 7. In LIGHT mode, as the light shining on the sensor gets brighter, more signal passes through it. In DARK mode, the signal increases as it gets darker. Furthermore, the light sensor has a sensitivity dial or slide dimmer to adjust how much light it takes to change the signal, and has a spectral sensitivity range from 500-to-600 nm similar to the LEDs wavelength. See Figure 10.

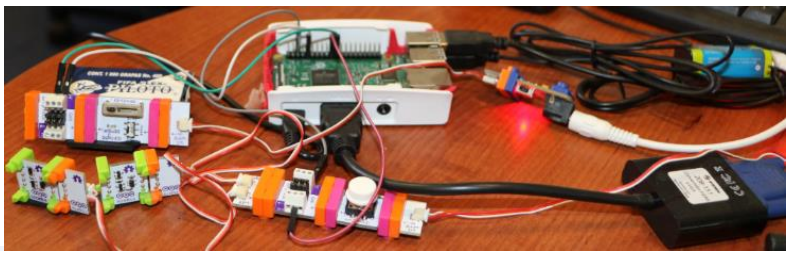

Fig. 10. Receiving a picture via VLC.

The recovered picture from the SPI MISO input is stored into a buffer, saved and displayed, as can be seen from the last 9 Python code lines of Figure 8.

4 Experiments and Results

For the experiments we use as was mentioned a RaspBerry Pi 3 computer with several LittleBits components. The whole setup is shown in Figure 11.

Figure 11 shows the RaspBerry Pi 3 computer connected through out bus connections to a HP display, keyboard and mouse. The figure also shows the recovered picture after the execution of the line command "***sudo python lena_spi.py***". Figure 12.

Figure 12 also shows that the recovered picture was not perfect. That was due to the presence of noise, mainly: Fluorescent light from ceiling lamps, misalignment and distance between LEDs and light sensor, and low sensitivity to light from the sensor.

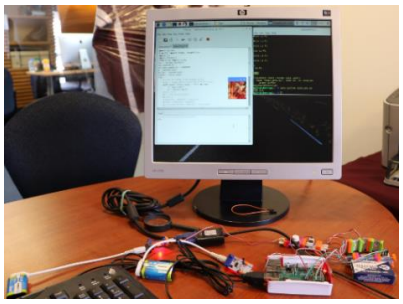

Fig. 11. Setup for the transmission and reception of images using VLC.

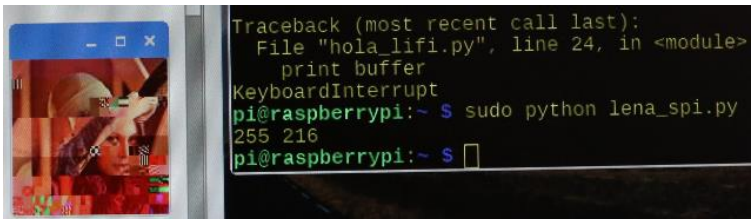

Fig. 12. Recovering of picture via VLC.

Fig. 13. Recovered picture via VLC after some adjustments.

4.1 Discussion of Results

After several adjustments to the setup and a few intents, the picture was finally well recovered. The whole transmission and recovering of a 225 x 225 pixel color image weighting 8 Kbytes, took less than a second. See Figure 13. Naturally, the noise cannot completely eliminated and increases when the misalignment and the distance between the LEDs and the light sensor is larger. Also the brightness of the LEDs influence the performance. That was the main reason for using two LEDs in parallel to increase the amount of light sent to the light sensor.

4.2 Metrics of the Recovered Picture via VLC

Because the transmission (LEDs) and reception (light sensor) link is made using Raspberry SPI, two questions arise: 1) What percentage of the image was recovered correctly? and 2) How many times the image was well recovered, versus the number of times the image could not be recovered?

1) Percentage of the image that was recovered correctly: As was shown in Figure 12, initially without any kind of adjustments, the recovery of the image was around 50 %. Notice from this figure the pixelation of the image in certain areas. In contrast with the adjustments mentioned before, the image was 100 % recovered as is shown in Figure 13.

2) Number of times the image was well recovered: Even though the adjustments performed to the transmitter-receiver setup, the link was shaky, requiring many tries to obtain a 100 % image recovery. The number of times the image was well recovered, was 3-out-of-10.

4.3 Contributions

This work differs with respect to similar works as in [8], [9] and [11], in the following. In [8] only a design model is proposed without any proof of implementation. In [9], a hardware setup is showed, but again, there is not any proof of sending and receiving any picture. It is also required to use two microcontrollers (one for the transmission and one for the reception). The main problem of using microcontrollers is that not all of them have enough RAM memory to store a medium to large image sent by the PC. Furthermore, the Matlab software have to reside at both PCs to connect with the microcontrollers and to process the picture downloading and uploading. In [11] a Windows PC with Matlab and an Arduino Uno microcontroller is used for the serial transmission and reception of two 24 x 25 color pictures, but without using visible light communication.

5 Conclusions

An image transmission and reception application using VLC was developed using a Raspberry Pi 3 computer, two bright LEDs and a light sensor from LittleBits, OOK modulation, and the Python Image Library. The picture in color, was recovered acceptably well although with a small presence of noise. It should be noted that this noise is due to environment light, and the distance between the LEDs and the light sensor.

Also, the performance of the application depends on the brightness of the LEDs and the sensitivity of the light sensor with respect to the light received from the LEDs and the surrounding light (in Figure 7 it can be seen that the light sensor has a control for graduating the sensitivity in the presence of high or low light). Additionally, the alignment between the LEDs and the light sensor influences the reception, and consequently the quality of the image reproduction.

6 Future Work

This application could be improved by: 1) Increasing the number and/or power of the LEDs to increase the reception distance; and 2) Implement and include a continuous

synchronization mechanism to improve the LED-to-light sensor VLC transmission-reception.

References

1. Haas, H.: Wireless data from every light bulb. In: TED Ideas worth spreading (2011)
2. Tsonev, Dobroslav, Videv, Stefan; Haas, H.: Light fidelity (Li-Fi): towards all-optical networking. In: Proc. SPIE (Broadband Access Communication Technologies VIII) 9007 (2) (2013)
3. Sherman, J.: How LED Light Bulbs could replace Wi-Fi. Digital Trends (2013)
4. Haas, H.: High-speed wireless networking using visible light. SPIE Newsroom (2013)
5. Vincent, J.: Li-Fi revolution: internet connections using light bulbs are 250 times faster than broadband (2013)
6. Wikipedia: Location awareness (2016)
7. Jovicic, A., Li, J., Richardson. T.: Visible light communication: opportunities, challenges and the path to market (2013)
8. Mahendran, R.: Integrated Lifi (Light Fidelity) For Smart Communication Through Illumination. In: International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 53–56 (2016)
9. Vyom S. et al: 2D Image Transmission using Light Fidelity Technology, Communication. International Journal of Innovation & Advancement in Computing Science, IJIACS, ISSN 2347-8616, Volume 4, Issue No. 4, pp. 121–126 (2015)
10. Kwonhyung, L., Hyuncheol. P.: Modulations for Visible Light Communications With Dimming Control. IEEE Photonics Technology Letters, Vol. 23, Issue 16 (2011)
11. Cubas Perfecto, G., Santiago Godoy, R., Anzueto Rios, A.: Transmisión de imágenes de Matlab a Arduino vía Puerto Serial. Boletín UPIITA-IPN, 644 CyT, No. 52 (2016)
12. LittleBits: https://www.littlebits.cc/ (2017)
13. Raspberry Pi: Raspberry Pi input and output pin voltage and current capability. Mosaic Documentation Web (2018)