1. INTRODUCTION

1.1 OVERVIEW

The visible light spectrum is the portion of the electromagnetic spectrum that is visible to the human eye. Electromagnetic radiation in this range of wavelengths is called visible light or simply light. A typical human eye will respond to wavelengths from about 390 to 700 nanometers. In terms of frequency, this corresponds to a band in the vicinity of 430–770 THz (Tera-hertz). Visible light communication is a method of data transfer using visible light spectrum. The visible light spectrum has a bandwidth spanning 100s of THz which is 10000 times larger than the RF bandwidth. VLC has numerous advantages over conventional methods, such as higher data rates, larger bandwidth, higher security, lower power consumption. Experiments done in VLC have achieved a data rate of 1Gbps under laboratory conditions. By using visible light, it is intended to achieve wireless communication in environments where it is not convenient to use Radio Frequency (RF) waves like in airplanes, hospitals etc. [4]. In VLC, the light source is typically a Light Emitting Diode (LED). Here LEDs are used for data transmission and offers illumination simultaneously. The main principle of this communication system is to switch ON and switch OFF a light source at a high rate, where the ON and OFF states of the light represent binary 1 and binary 0 respectively. By making the switching rate higher than the persistence of vision for humans, the light appears to be continuous in nature. Due to this continuous switching the overall brightness of the light decreases. The receiver works by looking out for the rapid changes of the light that represent data, thereby creating a line of sight communication link between the transmitter and the receiver.

VLC can potentially have a very large bandwidth but it is plagued by certain limitations owing to its physical components. At the transmitter, a white LED is used as a source of light. The limitation in bandwidth arises by the method in which the LED produces white light. At the receiver, the photodiode contains parasitic capacitances which limit the bandwidth of the VLC system. Ambient light interference introduces noise in the channel which affects system performance. The operating range is further limited by sensitivity of the photodiode. Having studied and understood the capabilities and limitations behind this technology, our research attempts to build a functional

visible light communication system to allow basic data transmission between two communicating nodes.

Figure 1.1 Representation of a VLC system

1.2 TOOL SET

1.2.1 Arduino Uno™

The Arduino Uno is an open source microcontroller based on the ATMega328p chip. It has a wide variety of on-board Input/output options including Digital I/O, Analog I/O and PWM outputs. The Arduino board is programmed using the Arduino IDE which is written in Java and based on other open source software. The board as well as the IDE was developed by Arduino.cc with a goal to provide an easy and efficient tool for prototyping. More information about the same can be found at [8].

Given its low cost and versatility it was the ideal choice for the project. In the project we use the Arduino Uno as an interface between the transmitting node and the light source and it helps to modulate the data onto the light. Another Arduino Uno is used at the receiver that act as an RS232 communication interface between the receiver front end circuit and the receiving node.

1.2.2 Python™

Python is a widely used general-purpose interpreted, objected-oriented, and a high-level programming language designed by Guido van Rossum and developed by Python Software Foundation. Python is user-friendly, easy to learn and has thousands of third-party modules and can integrate systems quickly, making it one of the most used programming languages among the diverse and international community of programmers and developers. A detailed explanation can be found at [9].

With its huge collection of libraries and interfaces, Python has proved to be the apt choice for image processing and file handling in our project. Python is used as a wrapper to control the flow of execution of programs including, reading data from files on PC, running shell commands for serial communication between the PC and the Arduino board, reading data from files and reconstruction of image from raw bit values using OpenCV [11].

1.2.3 OpenCV

OpenCV (Open Source Computer Vision), started by Intel in 1999, is the leading open source library for applications such as computer vision, image processing and machine learning. OpenCV is known for making use of multi-core processing for its computational efficiency, especially in real-time applications.

To transfer images through a VLC system, we need a powerful tool capable of converting images to raw pixel values to be encoded on the transmitter side and reconstruction of images from the received and decoded pixel values on the receiver side. OpenCV is the most suitable tool for our application, given its open source nature and online community support.

1.2.4 Processing

Processing is a software which focuses on writing code within the domain of the visual arts. It has a wide variety of APIs focusing on creating graphics. It also has APIs that allow for communication with the Arduino IDE.

Details about Processing can be found at [12]. It is this functionality that we looked to exploit in our project. We used processing to be able to select files from the computer and send it to the Arduino which would in turn send the file in the visible light channel. Though we didn't use processing in our final implementation it helped us to gain insights that were valuable in the progression of the project.

1.2.5 LTspice®

A Simulation Program with Integrated Circuit Emphasis or SPICE is a class of software used to simulate the characteristics of an electronic circuit. LTspice is a high-performance SPICE simulation software schematic capture and waveform viewer for the simulation of analog circuits [13]. We have extensively used LTspice for the design and simulations of our transmitter driver circuit as well as the receiver front end circuit. LTspice enabled us to effectively visualize the output from the circuit and as a tool to debug the circuit when errors were encountered.

1.2.6 C#

C# (pronounced as C sharp) is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust application using the .NET framework. More details about C# can be found at [15].

In this project C# was used to create a Windows application to transmit files to the Arduino's RS232 interface. The application called upon inbuilt APIs to do the same.

1.2.7 MATLAB

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment that was developed by MathWorks [16]. In our implementation we considered the use of the MATLAB image processing toolbox for the transmission of images through the visible light channel.

2. PROJECT EVOLUTION

In this chapter we look to throw some light on the efforts made over the course of this project. The development of the project was an iterative process and we have made many changes throughout this period to improve system performance.

The idea for this project came to us after watching a TED talk given by Prof. Harald Haas from the University of Edinburgh. The talk is what sparked an interest for us to pursue research in this domain. Exploring this domain led us to discovering a simple implementation of a VLC system on the Arduino platform. This work was done by a Piat Jonathan [1].

As a first step we replicated his work. This process also served as a proof of concept giving us the confidence to take this idea forward. The setup used was very simple, it consisted of two Arduino Uno boards. For the transmitter we used a single white LED. The led was connected to an output digital pin of the transmitter Arduino. The receiver hardware consisted of another white LED, but this was reverse biased so that we could exploit it as a photodiode. Given the simplicity of the hardware setup, the code was a little more involved. At the transmitter the data to be transmitted was converted to binary values, then Manchester encoded and put into a frame format. The data being transmitted was limited to 11 characters; this was because the data was being stored on the controller's limited memory. The frame format involved the addition of a preamble, a start of frame flag and an end of frame flag. The frame format i.e., the 11 bytes of character data and the added delimiters made the total size of the frame 38 bytes long. The frame was transmitted serially through a digital output pin on the controller, which was triggered by a timer interrupt. The LED was connected to this pin and was driven by the serial pulses of data. The receiver used the inbuilt 10 bit analog and digital converter (ADC) of the Arduino. The output of the photoreceiver at the receiver (LED in reverse bias) is a current proportional to the luminous intensity incident on it. As the ADC can't detect variations in current, we used a 1 MΩ resistance in parallel to the photoreceiver to convert the variations in current into voltage levels. This was an analog signal, so it was converted by the ADC on the microcontroller to signal variations into distinguishable digital pulses, that represent the transmitted data.

This implementation of the system allowed us to convince ourselves that data could be transmitted through visible light. We then wanted to tackle some of the limitations of the system such as the low data rate and the limited transmission distance.

To increase the transmission distance, we looked at improvements in the transmitter hardware. Our first iteration involved the use of a high brightness surface mount white led. This led had a higher current draw and hence needed a complimentary driver circuit. The driver circuit in this generation used a bipolar junction transistor as a switch. This gave an increase in transmission distance from a 1-2 cm to 15 cm.

Through some research we discovered that Metal Oxide Semiconductor Field Effect Transistors (MOSFET) can switch faster than BJTs. So, the second version of our hardware setup included the use of a MOSFET driver circuit. We used the IRFZ44n MOSFET [17] which is an N-channel MOSFET. After the implementation of this driver we expected an increase in the transmission speed of the system, this did not happen because we were being bottlenecked by the speed of the ADC at the receiver. Also as the MOSFET could easily handle high current loads on its output we switched our light source from the surface mount LED to a 12W micro-LED array. At the receiver we changed the photo sensing element from an LED in reverse bias to a specialized PIN Photodiode [14]. These two changes allowed for a huge improvement in the transmitted distance, from a distance of 15 cm we were now able to reliably communicate at a distance of 6 feet, which was thanks to the higher brightness of the LED array and the higher sensitivity of the photodiode.

As mentioned in the previous paragraph, we had still not managed to increase the transmission speed as we were bottlenecked by the computations occurring at the ADC. So, we looked to design a circuitry that could perform the analog to digital conversion but at a much higher rate. This led to the creation of our receiver front end circuity that we discuss in detail in upcoming chapters. The front-end circuit helped to greatly improve the speed of the system and increased the transmission speed of the system from 600 bps to 9600 bps.

The current implementation that we will discuss in the following chapters is able to transmit data at a maximum speed of 115200 bps and a distance of more than 8 feet, and the ability to transmit large text files and a limited selection of image files.

Achieving this was possible through an overhaul in the software system and minor optimizations of the hardware system.

3. Approach

In our approach to implement a VLC system, we have established communication between two computers. The basic principle is that the transmitter side is made to drive an LED connected to one of the digital output pins of the Arduino. The LED toggles ON and OFF based on the pulse received from the digital pin. The receiver uses a photodiode in reverse bias which produces current variations proportional to the received signal. These variations in current need to be further processed to obtain digital voltage signals that represent the data being transmitted.

3.1 Transmitter

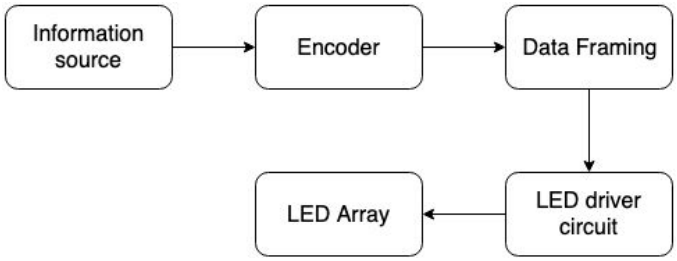

Figure 3.1 Block diagram for transmitter.

The basic idea behind a typical transmitter is to modulate the carrier in accordance to the message signal. here the carrier is light and message is the binary data, the carrier is modulated (i.e. switched ON & OFF) based on binary values. Data framing is taken care by serial library. We now look at how the transmitter was implemented in detail.

3.1.1 Transmitter Software

For the transmitter, feeding in serialized data isn't an ideal way to send the data across the visible light channel. This is because typically data consists of HIGH (1) and LOW (0) bits in which the number of 0s might be higher. This causes two problems: the first is that due to an increase in the number of transitions in the state of the led, there is a discernable flicker from the source which is not desirable; secondly, the overall brightness of the source drops below 50% as the time that the LED is ON is less than half of the total time. An encoding format is used to overcome the same. A modest way to encode the data is to use Manchester encoding, in which each data bit is encoded as a transition from low to high, or high to low, for an equal duration of time. Here a

bit valued ‘1’ is encoded as a signal going from HIGH state to LOW state and a ‘0’ is encoded to a signal going from LOW state to HIGH state. The Manchester encoded data prevents the reception of consecutive 0s. As a result, the LED brightness is not adversely affected. The encoding scheme used also helps reduce flickering and it is not visible to the naked eye.

Since we use asynchronous serial communication, framing is essential for reliable communication. Framing typically involves the addition of two special flags called delimiters that are added to the start and the end of a data segment. In our implementation we have used 0x03 and 0x02 for the start and stop bits respectively. Additionally, asynchronous communication requires data synchronization between the sender and receiver and this is achieved by a synchronization byte 0xD5 [1].

In order to transfer files from a computer to the microcontroller, a software overlay is needed to read from the file and push the data onto the Arduino through an RS232 interface. Probing into the possible solutions has resulted in exploring multiple programming language options including C++, python, java and C# and software including MATLAB, Processing and Microsoft Visual Studio. The selection of software depends on the trade-off between speed, efficiency and ease of use. While python offers diverse ready to use APIs, C++ promises speed and ease of bit manipulation. Similarly, reading various files from the computer proves to be easier on C# (run on Visual Studio) than on Processing (based on Java).

Since we use asynchronous serial communication, framing is essential for reliable communication. In our implementation we use the serial protocol for handling the data and to ensure effective communication; it takes care of data framing, formatting, and synchronization. We need two serial ports in our microcontroller one for receiving raw data from the computer and other to drive the LED driver circuit, but Arduino Uno has only one hardware serial port, to tackle this we use the *SoftwareSerial* library [10].

The Software Serial library has been developed by Arduino community to extend serial communication on other digital pins of the Arduino, using software to mimic the functionality (hence the name "*SoftwareSerial*"). It is possible to have multiple software serial ports with speeds up to 115200 bps.

3.1.2 Transmitter Hardware

The principle of data transmission in VLC is to turn ON and turn OFF a light source in correspondence to the transmitted bits. The transmitter in our system consists of a high-power white LED (12 Watts) and a MOSFET switch. This is interfaced to a computer (data source) using an Arduino Uno board that provides the RS-232 interface. Data from the computer is sent to the Arduino through UART where it is processed and encoded. The processed data is sent to the driver circuit through a digital output pin in the form of bits.

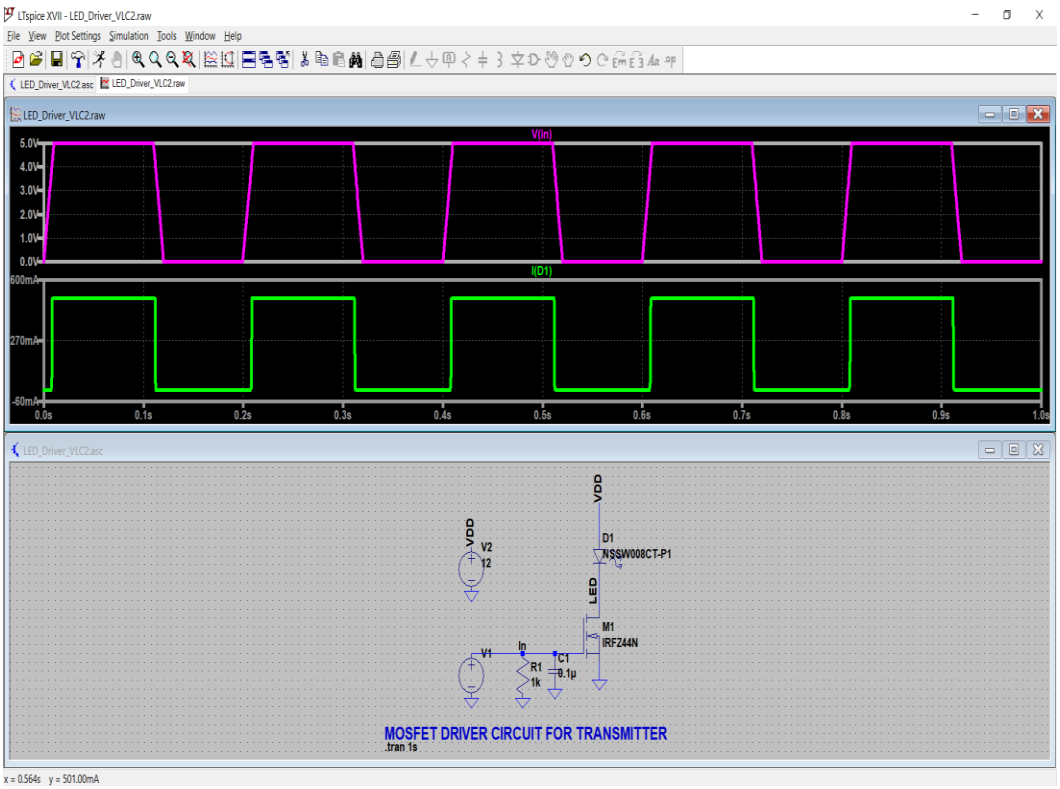

Figure 3.2 LTspice Simulation of MOSFET driver circuit.

The simulation results are as shown in figure 2.2. The plot in pink represents the input voltage pulses while the plot in green represents the current draw by the LED. As per the design the LED turns on whenever the input pulse is high.

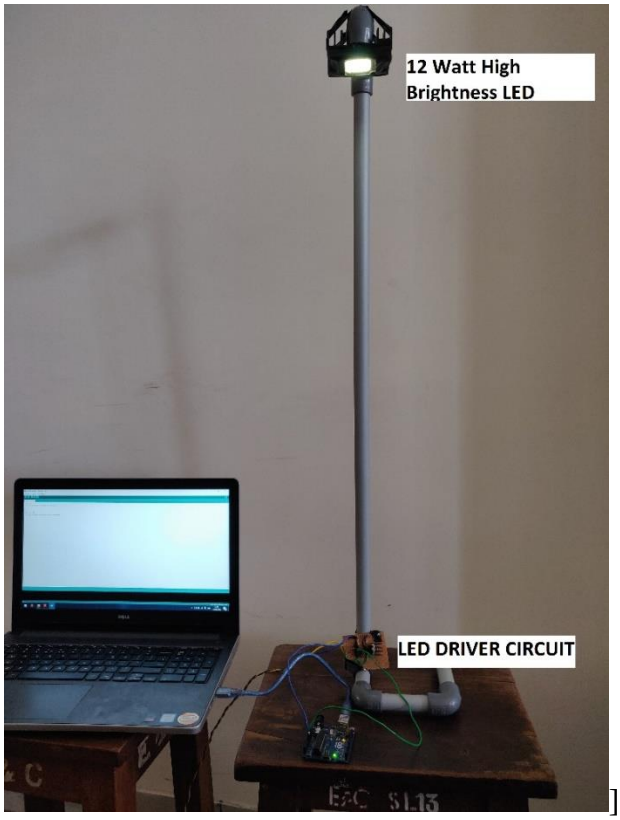

Figure 3.3 Experimental setup of Transmitter; the setup consists of High-Power LED source, LED driver circuit and micro-controller.

The driver circuit for the transmitter uses a MOSFET switch. We use an N-channel MOSFET with a typical threshold voltage (Vtn) of 4V. The logic levels of the microcontroller are 0-5V. Which means that the logic high output of the Arduino can fully turn on the MOSFET. The MOSFET circuit is used to rapidly switch the LED ON and OFF corresponding to the data output by the controller to the gate of the MOSFET. In the figure voltage V1 represents the output data bits from the controller and D1 represents the LED source that is being switched.

3.2 Receiver

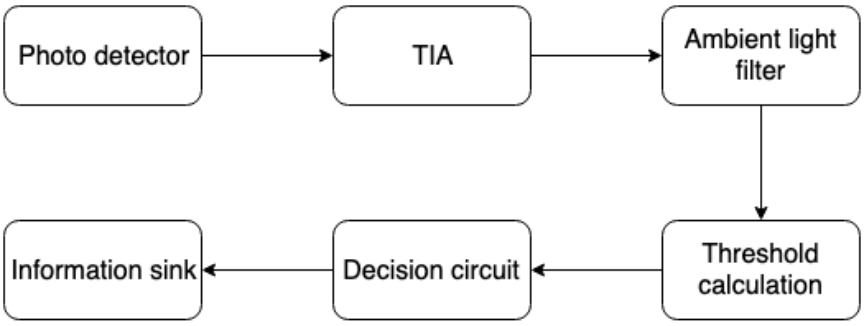

Figure 3.4 Block diagram of Receiver

As in any typical receiver circuit the process of reception begins with signal sensing. Since we are transmitting light, we need to detect the same, so the photo detector is the device that enables us to sense variations in the light intensity and do further signal processing accordingly. The received pixel values is written into a text file as a NumPy list by a python script. The text file consisting of a matrix of pixel values is read using the OpenCV library. The pixel values being a greyscale image have values in the range 0-255 which are then reconstructed into an image. In this section we discuss our approach in the development of our receiver setup.

3.2.1 Receiver Software

The digital signals from the receiver front end circuit are read by the digital pins of the Arduino. The signal being a digital PWM wave, it is necessary to identify the rising and falling edges of the signal to differentiate HIGH (1) and LOW (0) levels to convert the digital signals into a bit stream. To achieve this, the received digital signals are sampled at an appropriate frequency and a symbol period is calculated. If the signal is found to be steady for longer than a single symbol period, the data is considered valid and the receiver starts receiving.

The continuous stream of alternating 1s and 0s sent by the transmitter to keep the LED ON is ignored and the receiver only starts reading data when it encounters a synchronization byte. The receiver then removes the start

delimiter and extracts data until the stop delimiter is reached. The buffer limit of 64 bytes on an Arduino board is hard coded into its firmware source code. Larger data sent to through the buffer gets truncated, thus imposing a limiting factor on the size of each frame.

The data extracted from each frame is decoded from its Manchester form and coalesced into groups of 8 bits which represents a character. The Arduino then outputs the string of characters to the computer through its communication port.

3.2.2 Receiver Hardware

At the receiving end, the photo-sensing element used is a PIN Photodiode which can sense high frequency variations in the intensity of light being transmitted. The photodiode produces electric current which is proportional to the intensity of light incident on it. This electric current is in the order of few micro-amperes and analog in nature. The micro-controller can only recognize digital voltage levels which necessitate the need for a frontend processing circuit.

The frontend processing circuit should be able to convert electric current to equivalent voltage levels, filter the ambient interference, calculate a reference and output the digital signals.

To decide on efficient receiver hardware, we have simulated and tested the following frontend circuits: DC restoration circuit [7], AC coupled circuit and AC amplifier circuit. After a thorough analysis of simulation results, we have concluded that the AC coupled circuit is more reliable and efficient.

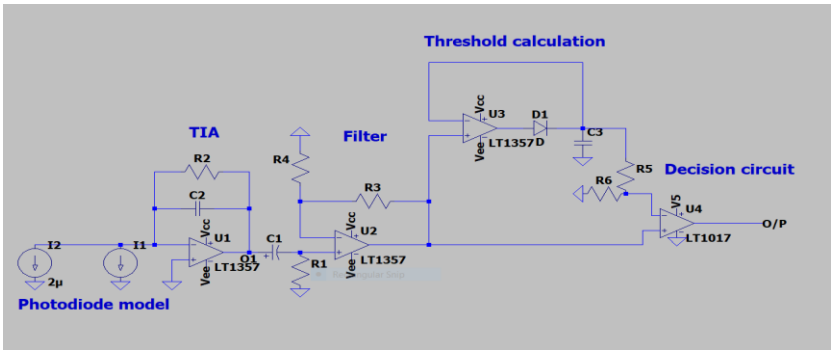

Figure 3.5 Simulation of the frontend processing circuit

The first stage in the processing circuit is a Trans-Impedance Amplifier (TIA) which converts electric current into voltage levels. The design of TIA is very crucial as it sets the upper limit of bandwidth. It also plays a significant role in reducing the effects of junction capacitance of the photodiode. Second stage involves removal of ambient light interference. The ambient light produces a DC shift in the received signal which is eliminated by AC coupling. Third stage involves calculation of a suitable voltage level which is set as reference for the comparator. Last stage is the digitization of voltage levels which is achieved by using a comparator. The comparator compares signals from second stage with the reference and outputs digital voltage levels to be read by the micro-controller.

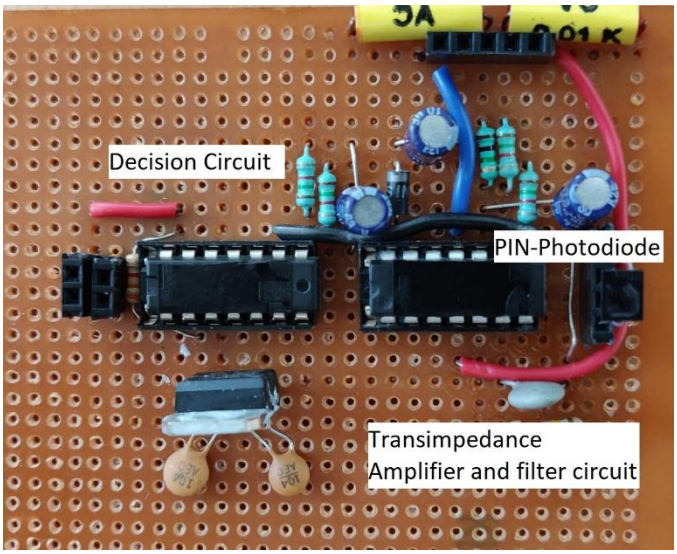

Figure 3.6 Frontend signal processing circuit for receiver; consists of PIN Photodiode as photo-detector, TIA stage, filter stage and decision stage. This circuit eliminates the need for a conventional ADC.

4. Choosing the right software

In this chapter we talk about the why behind the how, that is why we chose the toolset that we have used in our implementation amongst the viable options.

4.1 Arduino vs Raspberry Pi

Upon comparison of a microcontroller (Arduino) and a microprocessor (Raspberry Pi) for front-end data processing, Arduino proves to be a better fit for our application. Raspbian is a Linux based OS used on raspberry pi which provides easier operation of GPIO pins in python than in embedded C. In contrast, coding in embedded C for Arduino is simpler in terms of bit manipulation required for data encoding. Also, Raspberry pi being a full fledged independent system is more expensive than Arduino. With the use of an Arduino board, the computation intensive operations are carried on PCs which is much more powerful than a raspberry pi processor, is more versatile and can be scaled easily.

4.2 Processing vs C# vs Windows Shell commands

Arduino IDE is based on Processing and hence has a large collection of libraries and APIs to facilitate file handling between a PC and an Arduino. Processing offers ease of file transfer using the serial communication protocol. However, the Arduino has a 64-byte buffer which is hard coded into its kernel. This limitation in the buffer size restricts the size of data that can be sent. The data sent gets lost if it is greater than 64 bytes due to errors in framing. C# is more powerful compared to Processing but suffers with the same buffer limitation. Shell commands can be used on the PC to send data to Arduino through the serial port. It automatically takes care of framing. Since data is packetized, with each packet being less than 64Kb, the Arduino is able to receive data without loss.

4.3 Python vs MATLAB

MATLAB provides easy manipulation of image files. Images can be directly read or written into a matrix using inbuilt functions. However, MATLAB is inefficient when it comes to running shell commands from the program. Python has an OpenCV library which can be used for image manipulation and OS libraries to run shell commands from the python scripts. Python having a huge online support community and an easy programming interface makes it a better choice for our application.

5. Results and Observations

The proposed system is capable of taking either a string or a text file as input. The data is then encoded and framed using the above-mentioned frame format. The frames are fed bitwise to the LED driver circuit. The system is tested for reliable performance at a communication speed of 600bps with the transmitter and receiver separated by a distance of 6ft.

Initially, the receiver used the inbuilt Arduino ADC instead of a frontend circuit for signal conversion. Although the system was simple and reliable, the computationally intensive 10-bit ADC consumed a greater number of clock cycles and slowed down the communication speed. To overcome this limitation, the aforementioned receiver circuit was built and implemented.

5.1 Simulation of frontend circuit

Before realizing the frontend receiver hardware, an LTSpice model of the circuit was built and analyzed. Simulation was tested based on scenarios that the system was expected to encounter such as ambient light interference. The expected DC shift from the ambient light was seen at the output of the TIA. The filter stages and decision circuitry performed as designed and removed this shift to yield a near perfect digital signal as shown in Figure 3.1.

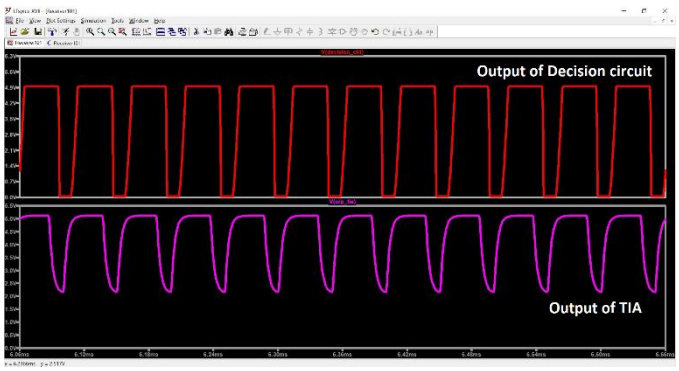

Figure 5.1 Simulation results of the frontend circuitry. Lower waveform shows the output of the TIA, with the DC shift from the ambient light interference. Upper waveform is the final expected output of the circuit.

5.2 Comparison of Simulation with Actual Results

The receiver frontend circuit was tested under various ambient lighting conditions. It was also tested at different separations and misalignments and was found to be reliable with no data loss or corruption.

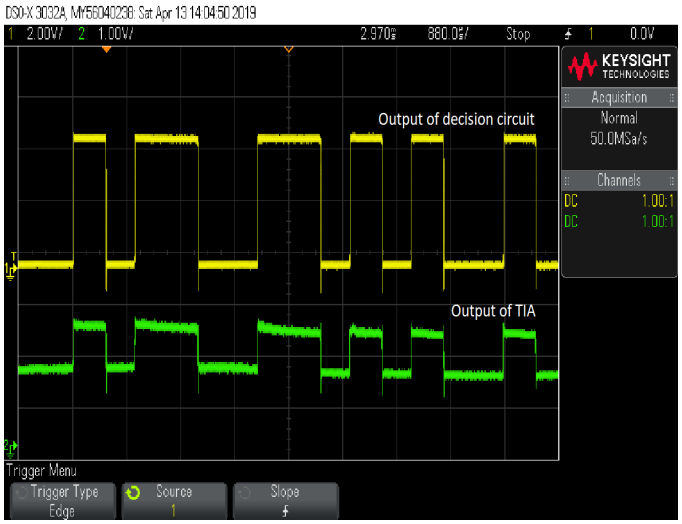

Figure 5.2 Output waveforms of frontend receiver circuit; top waveform indicating output of TIA and bottom waveform indicating output of decision circuit.

In Figure 6, we can see that the output of TIA is an analog signal with a DC shift of 1.5 volts. This signal is filtered and fed to the decision circuit whose output is represented by the upper waveform. We found that the actual results met the expectation set by the simulations.

5.3 Observed Outputs and Analysis

The first phase of the project was to establish a reliable communication system which could send the phrase "Hello NIECEn" without any errors (n represents the loop counter). Initial hardware setup used a BJT driver circuit and an LED on the receiver side working as a photodetector. The separation between the transmitter and receiver was only about 15cm. The Arduino code used the inbuilt ADC to sample and differentiate 0s and 1s. The system could successfully transmit data at 600bps, but was limited to a size of 16 bytes.

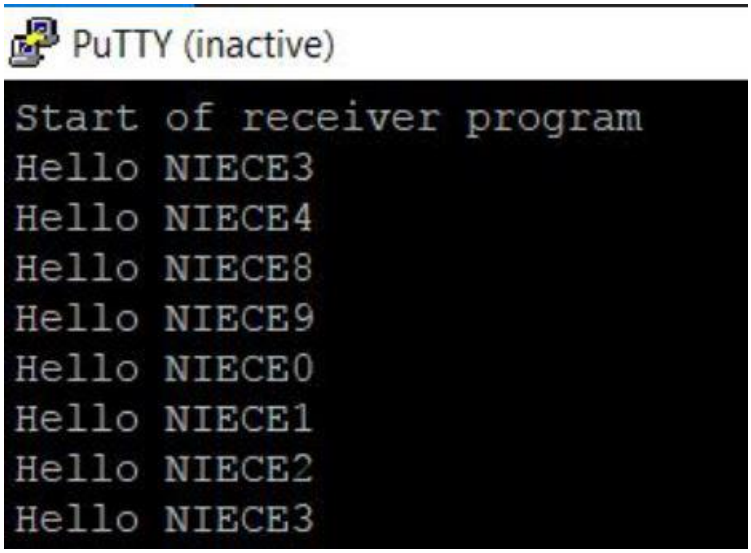

Figure 5.3 Output of project first phase. Displaying "Hello NIECEn".

The next phase of the project saw huge improvements in terms of distance which increased to 6 meters, and data size increased to 64 bytes. The increase in transmission distance was a result of the use of a specialized photodiode [14]. The improvements in reliability were due to a faster switching action at the transmitter side now using a MOSFET driver circuit. The data to be transmitted could now be input at runtime from the serial window, but was limited by the buffer size of the Arduino board.

The final phase of the project presents an efficient system with no limits on text size. The receiver is a three-stage circuit which replaces the inbuilt Arduino ADC, making it faster. A python script is used to run the shell commands for data transfer from the PC to Arduino, and OpenCV for image and text file handling. Transmission of text files and image files is now possible without any loss or errors. The system now uses standard serial communication protocols to transmit data. Text files of 1Mb has been sent successfully at a speed of 9600bps, about 15 times the original speed. Uncompressed image files of size 32x32 pixels can be transmitted.

Figure 5.4 32x32 image to be transmitted

VLC - Notepad

File Edit Format View Help

```
25 20 27 21 38 23 24 15 20 15 4 32 7 1 53 114 132 171 175 110 69 42 1 10 15 28 27 45 30 37 39 46
24 20 34 15 26 25 21 18 19 16 12 14 5 72 157 213 243 248 241 242 244 165 48 7 3 38 37 37 28 35 30 45
21 20 35 12 16 25 21 18 11 13 22 5 24 159 236 255 246 252 255 250 249 250 164 8 0 41 45 29 28 36 25 48
18 22 23 22 23 23 28 17 11 11 23 3 61 221 250 244 255 248 252 255 236 254 242 124 6 32 46 31 30 38 31 52
19 26 11 43 44 28 41 17 22 15 16 0 90 248 245 243 244 254 245 255 255 235 226 197 52 23 44 41 32 38 37 52
23 28 15 60 58 40 44 19 21 17 14 4 101 245 235 247 236 255 253 246 251 255 244 208 110 24 44 48 34 34 38 48
24 20 33 62 52 54 32 19 14 17 16 10 96 232 230 233 255 230 244 253 242 245 239 218 153 34 45 47 35 33 37 51
23 11 49 56 39 63 17 16 17 18 14 7 84 227 237 225 223 233 254 249 252 248 221 231 171 41 46 42 36 34 37 57
17 19 54 50 44 49 15 20 16 9 25 0 98 215 183 185 191 214 229 226 237 233 205 204 196 72 39 59 40 32 43 49
16 23 60 50 44 48 18 20 16 3 21 10 57 139 117 72 55 60 147 221 170 90 75 85 126 64 31 50 55 35 37 59
12 29 68 50 42 44 21 19 10 0 9 18 42 106 117 91 50 25 102 206 140 23 25 56 77 60 25 45 61 33 37 62
9 33 74 47 38 36 22 16 10 13 12 11 25 59 54 76 153 144 146 216 208 107 78 91 116 83 26 43 54 27 44 55
7 34 75 42 33 26 21 14 0 5 12 4 41 130 138 188 207 230 188 217 255 228 192 191 203 129 30 34 52 27 41 53
8 34 74 36 30 16 19 12 24 12 16 0 19 176 214 232 226 236 192 195 234 224 212 216 248 151 27 19 56 35 32 58
13 34 71 33 29 9 18 12 6 1 9 6 19 184 243 220 235 212 202 228 253 244 230 234 218 111 16 14 46 40 34 55
16 34 69 31 29 5 18 13 10 12 1 12 16 164 247 230 240 182 176 220 255 248 223 228 168 53 5 19 29 40 45 45
14 52 62 30 19 19 5 10 15 1 7 4 26 132 227 237 194 157 170 178 230 213 205 204 152 55 3 25 22 40 38 50
18 49 60 31 14 12 3 9 0 4 12 0 22 137 209 178 181 131 65 42 81 115 166 173 166 69 5 18 25 44 34 40
26 42 53 30 8 6 3 8 6 6 5 20 68 139 179 171 208 202 129 92 74 132 187 145 166 71 9 17 30 45 33 35
33 33 41 24 6 7 5 6 7 7 0 38 110 130 137 163 220 228 198 205 171 214 203 121 156 54 8 23 31 42 40 38
38 27 29 16 7 13 9 6 4 15 11 59 131 151 139 126 153 146 117 119 96 130 134 157 157 38 4 27 31 37 47 37
37 29 24 10 10 18 9 7 6 5 19 87 152 183 178 128 100 135 126 100 100 111 100 150 150 32 3 23 33 38 48 33
34 36 28 7 11 18 7 9 15 8 7 72 139 166 178 157 158 160 129 88 106 121 136 130 99 22 9 20 42 40 43 36
31 42 33 7 10 16 3 11 39 58 18 36 107 134 153 172 141 125 146 149 123 89 156 131 37 10 17 20 50 42 39 44
39 35 25 6 6 10 1 7 13 79 74 16 60 127 139 181 189 206 214 212 191 203 201 113 12 18 10 24 41 40 44 38
45 42 32 7 4 9 2 7 0 25 61 34 18 52 88 131 151 190 181 170 165 145 102 32 7 20 14 22 36 41 43 33
43 43 34 6 0 6 2 6 13 0 25 39 33 45 41 30 71 89 93 85 60 57 62 18 0 17 18 20 34 44 46 34
35 37 33 6 0 4 1 2 1 0 0 2 23 48 48 53 45 15 0 10 17 46 62 16 0 17 22 24 36 44 42 36
35 35 35 13 0 5 0 0 5 12 5 0 0 10 12 30 68 66 36 34 57 47 17 0 11 18 24 29 39 41 35 36
39 34 36 20 6 8 1 0 6 0 1 8 10 6 0 0 0 17 26 25 17 1 0 5 8 10 18 26 39 44 37 43
33 20 23 15 5 6 2 2 10 4 7 8 0 0 5 13 8 0 2 6 0 0 10 3 2 4 13 16 33 47 38 40
21 3 6 5 0 2 2 6 0 6 0 0 7 9 4 1 0 12 5 0 11 4 0 14 6 10 17 9 24 43 30 25
```

Figure 5.5 matrix of 32x32 image to be transmitted

COM3

```
25 20 27 21 38 23 24 15 20 15 4 32 7 1 53 114 132 171 175 110 69 42 1 10 15 28 27 45 30 37 39 46
24 20 34 15 26 25 21 18 19 16 12 14 5 72 157 213 243 248 241 242 244 165 48 7 3 38 37 37 28 35 30 45
21 20 35 12 16 25 21 18 11 13 22 5 24 159 236 255 246 252 255 250 249 250 164 8 0 41 45 29 28 36 25 48
18 22 23 22 23 23 28 17 11 11 23 3 61 221 250 244 255 248 252 255 236 254 242 124 6 32 46 31 30 38 31 52
19 26 11 43 44 28 41 17 22 15 16 0 90 248 245 243 244 254 245 255 255 235 226 197 52 23 44 41 32 38 37 52
23 28 15 60 58 40 44 19 21 17 14 4 101 245 235 247 236 255 253 246 251 255 244 208 110 24 44 48 34 34 38 48
24 20 33 62 52 54 32 19 14 17 16 10 96 232 230 233 255 230 244 253 242 245 239 218 153 34 45 47 35 33 37 51
23 11 49 56 39 63 17 16 17 18 14 7 84 227 237 225 223 233 254 249 252 248 221 231 171 41 46 42 36 34 37 57
17 19 54 50 44 49 15 20 16 9 25 0 98 215 183 185 191 214 229 226 237 233 205 204 196 72 39 59 40 32 43 49
16 23 60 50 44 48 18 20 16 3 21 10 57 139 117 72 55 60 147 221 170 90 75 85 126 64 31 50 55 35 37 59
12 29 68 50 42 44 21 19 10 0 9 18 42 106 117 91 50 25 102 206 140 23 25 56 77 60 25 45 61 33 37 62
9 33 74 47 38 36 22 16 10 13 12 11 25 59 54 76 153 144 146 216 208 107 78 91 116 83 26 43 54 27 44 55
7 34 75 42 33 26 21 14 0 5 12 4 41 130 138 188 207 230 188 217 255 228 192 191 203 129 30 34 52 27 41 53
8 34 74 36 30 16 19 12 24 12 16 0 19 176 214 232 226 236 192 195 234 224 212 216 248 151 27 19 56 35 32 58
13 34 71 33 29 9 18 12 6 1 9 6 19 184 243 220 235 212 202 228 253 244 230 234 218 111 16 14 46 40 34 55
16 34 69 31 29 5 18 13 10 12 1 12 16 164 247 230 240 182 176 220 255 248 223 228 168 53 5 19 29 40 45 45
14 52 62 30 19 19 5 10 15 1 7 4 26 132 227 237 194 157 170 178 230 213 205 204 152 55 3 25 22 40 38 50
18 49 60 31 14 12 3 9 0 4 12 0 22 137 209 178 181 131 65 42 81 115 166 173 166 69 5 18 25 44 34 40
26 42 53 30 8 6 3 8 6 6 5 20 68 139 179 171 208 202 129 92 74 132 187 145 166 71 9 17 30 45 33 35
33 33 41 24 6 7 5 6 7 7 0 38 110 130 137 163 220 228 198 205 171 214 203 121 156 54 8 23 31 42 40 38
38 27 29 16 7 13 9 6 4 15 11 59 131 151 139 126 153 146 117 119 96 130 134 157 157 38 4 27 31 37 47 37
37 29 24 10 10 18 9 7 6 5 19 87 152 183 178 128 100 135 126 100 100 111 100 150 150 32 3 23 33 38 48 33
34 36 28 7 11 18 7 9 15 8 7 72 139 166 178 157 158 160 129 88 106 121 136 130 99 22 9 20 42 40 43 36
31 42 33 7 10 16 3 11 39 58 18 36 107 134 153 172 141 125 146 149 123 89 156 131 37 10 17 20 50 42 39 44
39 35 25 6 6 10 1 7 13 79 74 16 60 127 139 181 189 206 214 212 191 203 201 113 12 18 10 24 41 40 44 38
45 42 32 7 4 9 2 7 0 25 61 34 18 52 88 131 151 190 181 170 165 145 102 32 7 20 14 22 36 41 43 33
43 43 34 6 0 6 2 6 13 0 25 39 33 45 41 30 71 89 93 85 60 57 62 18 0 17 18 20 34 44 46 34
35 37 33 6 0 4 1 2 1 0 0 2 23 48 48 53 45 15 0 10 17 46 62 16 0 17 22 24 36 44 42 36
35 35 35 13 0 5 0 0 5 12 5 0 0 10 12 30 68 66 36 34 57 47 17 0 11 18 24 29 39 41 35 36
39 34 36 20 6 8 1 0 6 0 1 8 10 6 0 0 0 17 26 25 17 1 0 5 8 10 18 26 39 44 37 43
33 20 23 15 5 6 2 2 10 4 7 8 0 0 5 13 8 0 2 6 0 0 10 3 2 4 13 16 33 47 38 40
21 3 6 5 0 2 2 6 0 6 0 0 7 9 4 1 0 12 5 0 11 4 0 14 6 10 17 9 24 43 30 25
```

Figure 5.6 matrix of 32x32 image received

Figure 5.7 reconstructed 32x32 image

However, an attempt to transmit a 64x64 image results in concatenation of a few pixel values in a row, making it unable to reconstruct the original image back from these values. These errors occur at random positions and manual correction of these values is required to reconstruct the image.

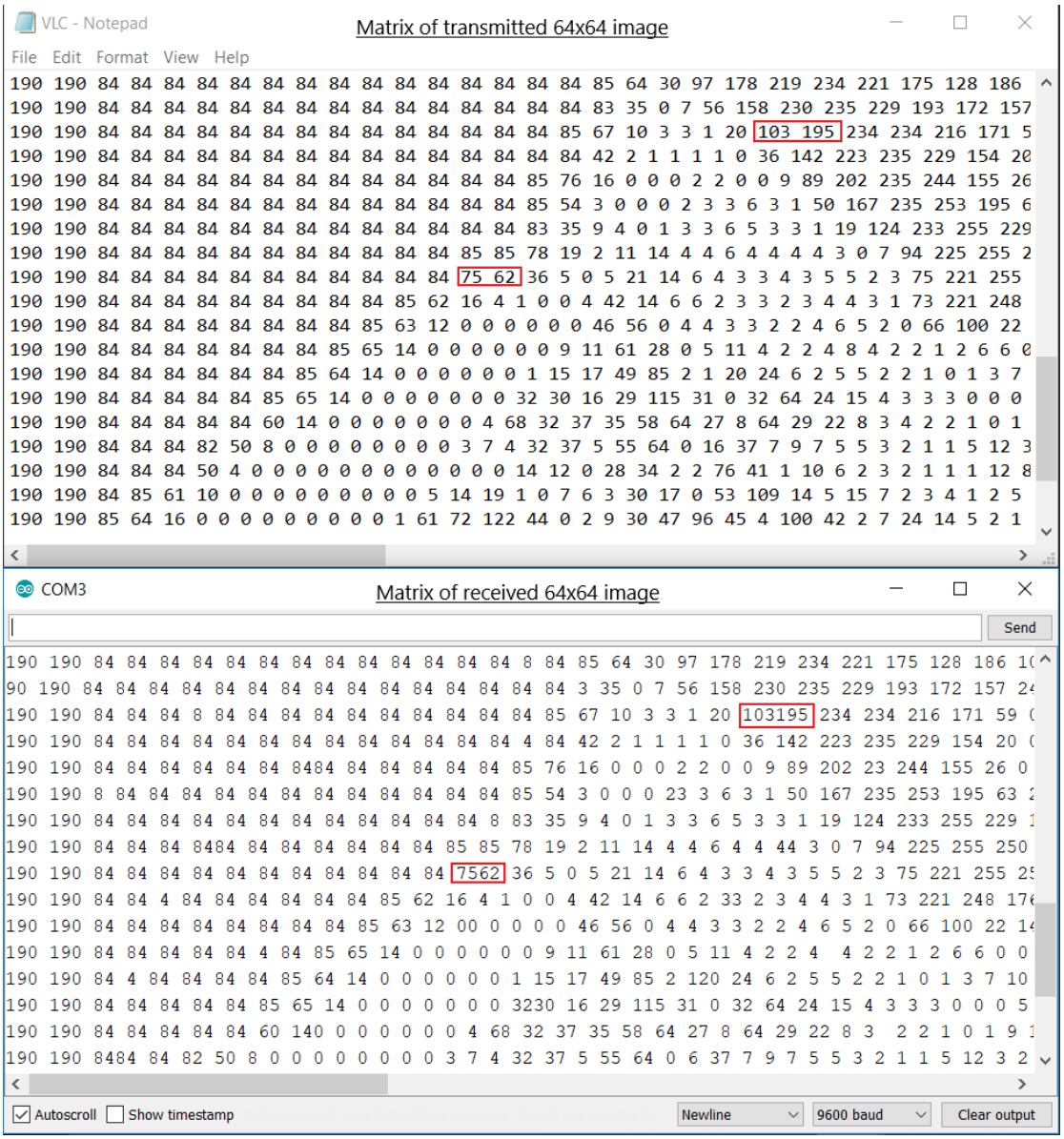

Figure 5.8 matrix of 64x64 image transfer, the upper image shows transmitted matrix and lower image shows received matrix, we can see missing spaces at random places which make it impossible to reconstruct the image.

6. LIMITATIONS

Though ideally it seems that VLC has a very large bandwidth and can be a viable solution for the dearth of bandwidth that exists today, VLC has its share of limitations. Its bandwidth is limited by the typical elements involved which are LEDs and Photodiodes. At transmitter side a white LED is used as a source of light, the limitation in bandwidth arises by the method in which a LED produces white light.

White light is a mixture of Red, Green and Blue light. The RGB white method produces white light by combining the output from red, green and blue LEDs. This is an additive color method. But RGB white light is hardware-intensive, since it requires three LEDs, and it tends to render pastel colors unnaturally, a fact which is largely responsible for the poor color rendering index of RGB white light. The Phosphor white method produces white light in a single LED by combining a short wavelength LED such as blue or UV, and a yellow phosphor coating [6]. It is seen that the bandwidth is ~2MHz for the white, and ~10MHz for the blue component only [2]. This is due to the long decay time of the phosphor, and provides a limitation on the overall bandwidth available. In addition, the Blue LED die is not designed for high speed operation and is very large in area (and thus has high equivalent capacitance) compared with devices used for high speed communications. The limited bandwidth of the LEDs is therefore one of the major challenges for high-speed communications.

At the receiver, studying a model of the photodiode led to the discovery of a junction capacitance which adversely affects the bandwidth of the VLC system. Ambient light interference introduces noise in the channel which if large enough, saturates the photodiode current. The operating range is also limited by the sensitivity of the photodiode. Factors limiting the performance of the photodiode are the capacitance of the junction as well as the parasitic capacitance of the package. The reverse-biased PIN junction acts like a capacitor because charge can be stored in the P and N sides with the depletion layer acting as a dielectric. This junction capacitance is to be minimized by careful design of the diode structure for high speed performance. Most practical high speed diodes

typically have sub Pico-farad capacitances. Minimizing the load resistance also improves the performance of the receiver as a whole but it is not a limitation of the diode itself, but is related to the receiver's electrical design.

The transmitter circuit uses ON OFF Keying (OOK) to modulate the light source at a high frequency. However, OOK is more sensitive to additive noise, has a poor spectral efficiency and power efficiency. The Manchester encoding used to have an equal number of ones and zeros offers only 50% efficiency.

7. FUTURE WORK

VLC requires the light source to be always ON. A robust transmitter can be designed which is capable of adapting it's brightness by varying the intensity based on received signal strength and ambient conditions. A better substitution for Manchester encoding which has more number of 1s can increase the efficiency and also keep the light source ON for a longer duration. Alternate modulation schemes such as FSK and BPSK which provide better immunity to noise can be implemented. To obtain gigabit speeds and parallel transmissions, OFDM and optical MIMO techniques can be used [2].

In spite of having a robust receiver, there might be few errors in the data received. Hence performance of code has to be improved. This can be done by using better encoding techniques and including data bits for error correction. Multiple photo detectors can be used in parallel to increase the current generated, thereby increasing the transmission distance. Solar panel has a very large surface area compared to a photodiode; hence they receive large optical power. They also receive illumination from ambient light; the DC offset produced by ambient light in photodiodes being small cannot be harnessed but in solar cells the DC offset is large enough to be harvested. In this way a self-sustained receiver can be developed [5].

8. APPLICATIONS

VLC has the following implicit characteristics including high bandwidth, low power consumption, inherent security and non-licensed channels that have made it beneficial for practical use. Diverse application scenarios are as follows: Every modern-day vehicle embodies LED headlights which can be used to implement VLC modules. Applications in this space include collision warning, lane change warning, traffic rule violation warning and pre-accident sensing. Devices and machines sensitive to electromagnetic interference (such as MRI scanners) can switch to VLC technology in hospitals to prevent interference with radio waves from other machines. Large volumes of data can be transferred with adequate security by establishing a connection between two mobiles pointed at each other. VLC can provide much higher data transfer rates than Wi-Fi or Bluetooth. VLC can be incorporated in passenger compartments of aircraft where radio is undesirable. LEDs in the cabin can also be used to provide multimedia streaming services and flight tracking details to passengers. RF signals do not work underwater. VLC devices can be used for short distance communication underwater between divers or submarines and the surface.

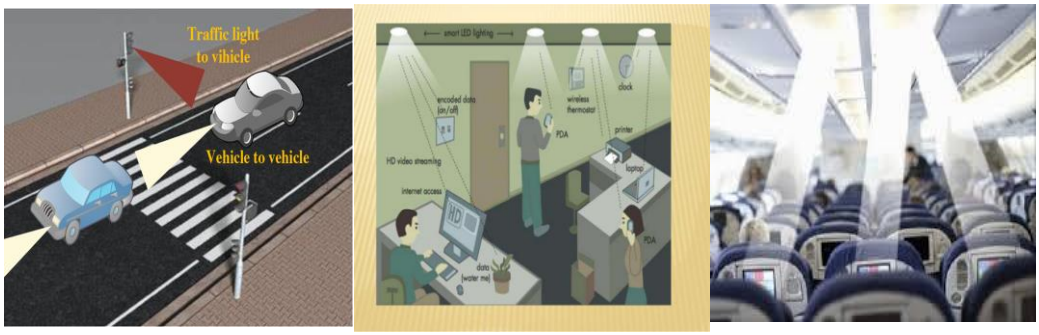

Figure 8.1 Illustration representing application of VLC in different scenarios.

9. REFERENCES

[1] Piat Jonathan, Arduino simple Visible Light Communication, https://github.com/jpiat/arduino/wiki/Arduino-simple-Visible-LightCommunication

[2] D. C. O'Brien, L. Zeng, H. Le-Minh, G. Faulkner, J. W. Wa lewski, and S. Randel, Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on, 2008. "Visible light communications: Challenges and possibilities," 2008.

[3] A. E. Kelly, J. J. D. McKendry, S. Zhang, D. Massoubre, B. R. Rae, R. P. Green, R. K. Henderson, and M. D. Dawson, "High -speed GaN micro-LED arrays for data communications," in Transparent Optical Networks (ICTON), 2012 14th International Conference on, 2012, pp. 15.

[4] H. Haas and C. Chen, European Conference on Optical Communication (ECOC), What is LiFi?, 2015

[5] Zixiong Wang, Dobroslav Tsonev, Stefan Videv, and Harald Haas, IEEE journal on selected areas in communications, VOL. 33, NO. 8, AUGUST 2015, "On the design of a solar panel receiver for Optical Wireless Communication and simultaneous energy harvesting", 2015

[6] Photon Star Technology Ltd., "How LEDs produce white light" http://www.photonstartechnology.com/learn/how_leds_produce_white_light

[7] Bonnie Baker, "Photo sensing with ambient background", February 17 2011. https://www.edn.com/Pdf/ViewPdf?contentItemId=4364018

[8] What is Arduino? https://www.arduino.cc/en/Guide/Introduction

[9] About Python, https://www.python.org/about

[10] SoftwareSerial Library, https://www.arduino.cc/en/Reference/SoftwareSerial

[11] OpenCV Wiki, https://github.com/opencv/opencv/wiki

[12] processing, https://github.com/processing/processing/wiki

[13] LTspice, Analog Devices, https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

[14] VEMD5510C Silicon Photodiode, Vishay Semiconductors, https://datasheet.lcsc.com/szlcsc/Vishay-Intertech-VEMD5510C_C191126.pdf

[15] Introduction to the C# Language and the .NET Framework, Microsoft, https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[16] MATLAB, MathWorks Inc., https://en.wikipedia.org/wiki/MATLAB

[17] IRFZ44 Datasheet, Vishay Semiconductors, https://www.vishay.com/docs/91291/91291.pdf