Santa Clara University
Scholar Commons

Electrical Engineering Senior Theses | Engineering Senior Theses

6-12-2018

Vehicle to Everything Communication using VLC

Andrew Harris
Santa Clara University, amharris@scu.edu

Michael Karachewski
Santa Clara University, mkarachewski@scu.edu

Nick Schnabel
Santa Clara University, nschnabel@scu.edu

Follow this and additional works at: https://scholarcommons.scu.edu/elec_senior

 Part of the Electrical and Computer Engineering Commons

Recommended Citation
Harris, Andrew; Karachewski, Michael; and Schnabel, Nick, "Vehicle to Everything Communication using VLC" (2018). *Electrical Engineering Senior Theses*. 44.
https://scholarcommons.scu.edu/elec_senior/44

This Thesis is brought to you for free and open access by the Engineering Senior Theses at Scholar Commons. It has been accepted for inclusion in Electrical Engineering Senior Theses by an authorized administrator of Scholar Commons. For more information, please contact rscroggin@scu.edu.

SANTA CLARA UNIVERSITY

Department of Electrical Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Andrew Harris, Michael Karachewski, Nick Schnabel

ENTITLED

Vehicle to Everything Communication using VLC

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
ELECTRICAL ENGINEERING

Sarah Harris

Thesis Advisor Signature

June 12, 2018
date

Shoba Krishnan

Department Chair Signature

6/13/18
date

SANTA CLARA UNIVERSITY
DEPARTMENT OF ELECTRICAL ENGINEERING

Vehicle to Everything Communication using VLC

by

Andrew Harris
Michael Karachewski
Nick Schnabel

Santa Clara, California
June 12, 2018

Vehicle to Everything Communication using VLC

Andrew Harris
Michael Karachewski
Nick Schnabel

Department of Electrical Engineering
Santa Clara University
June 12, 2018

ABSTRACT

Autonomous vehicles are currently self contained. However, we are coming to an age where there will be so many smart vehicles on the road, they will need a way to communicate with each other. In addition, the radio frequency spectrum is crowded and additional communication methods are needed to keep up with future needs. With these two ideas, this paper looks into a very new and developing form of communication - visible light communication - to provide an alternative to traditional radio waves for vehicular communication. We analyze the viability of this system as a proof-of-concept for the technology. We also present a design-prototype of a VLC vehicular transmitter and receiver to establish a wireless communication channel. We hope that our work can help provide a foundation for such a system being used in vehicles in a connected network.

Table of Contents

List of Figures

1 Introduction

1.1 Problem Statement

With self-driving cars just on the horizon, our modern world will be experiencing some major changes. Currently, self-driving cars sense their surroundings through a series of LIDAR sensors and cameras. These devices work completely internally to each respective vehicle, and do not provide the ability to pass data between autonomous vehicles. One of the main advantages of having self-driving cars is the increase in safety and efficiency of transportation. Along those lines, if there is one overarching system that connects all the vehicles, then the safety of driving can ultimately be improved even more while optimizing traffic times and navigation routes as well. To have such an overarching system, autonomous cars must be connected both to each other and to the system as a whole. We are trying to provide a system that can work in conjunction with current solutions, potentially providing the basis for a network of autonomous vehicles. Our system could potentially provide some of the infrastructure that would allow this interconnectivity to be developed.

As shown in Figure 1 below, the free space in the frequency spectrum for all normal communication specifications is very full. Because of this, it is necessary to move up into a relatively new area for communications: the visible light spectrum.

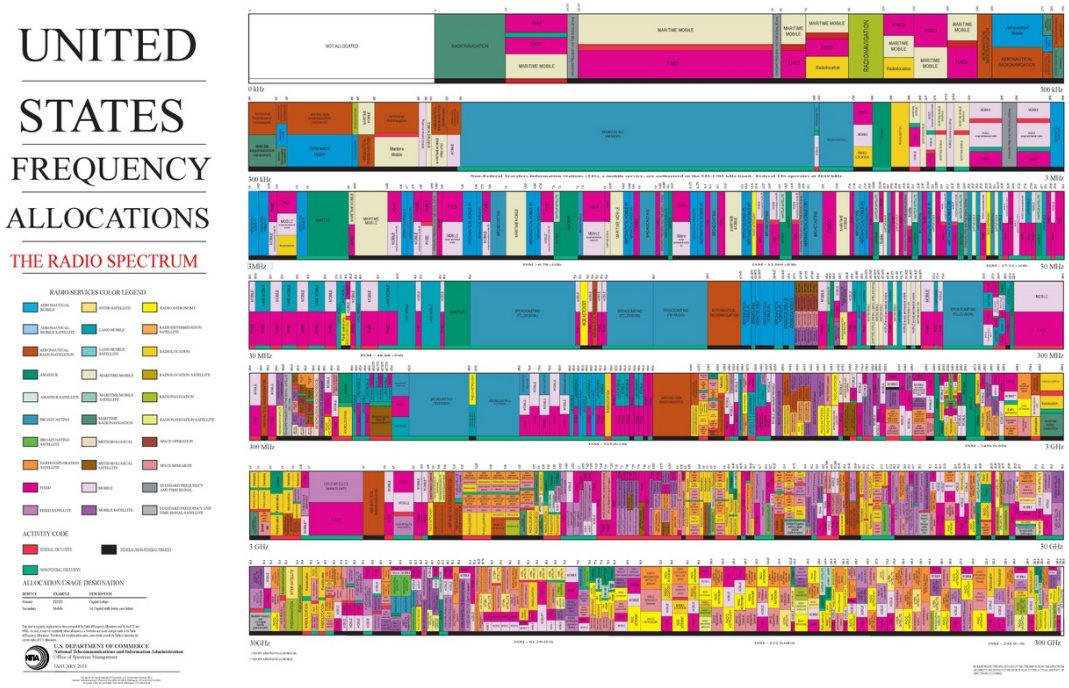

Figure 1: United States Frequency Allocation Chart [1]

1.2 Background information

Visible Light Communication is a form of wireless data communication that utilizes the visible light portion of the electromagnetic spectrum, between 400 and 800 THz, as can be seen in Figure 2 below. Because VLC can be implemented using traditional light sources such as LEDs that blink at a fast rate, information up to 500 Mbit/s can be transmitted without humans perceiving this blinking (it looks like a steady stream of light) [2]. Existing modulation techniques, such as OFDM and on/off keying, can be used to modulate visible light signals in order to send the data [3]. On the receiver side, a network of photodetectors, such as that in a cell phone camera, can demodulate the messages. The fact that such a communication system can be implemented through existing light sources such as car headlights or living room lamps, along with the fact that the RF spectrum is rapidly becoming filled, make VLC a lucrative and growing field of study. In addition, light based systems are confined by walls and dont interfere with existing RF signals, and thus can lead to enhanced security and re-usability [4]. With regards to connected cars, it becomes clear how VLC could be implemented through car headlights and tail-lights as well as street signs and stoplights. Currently, technology for Car to Everything (V2X) communication is in the early stages, but companies like Qualcomm are exploring cellular solutions through their C-V2X chipset, which they see as paving the way toward 5G technology [5].

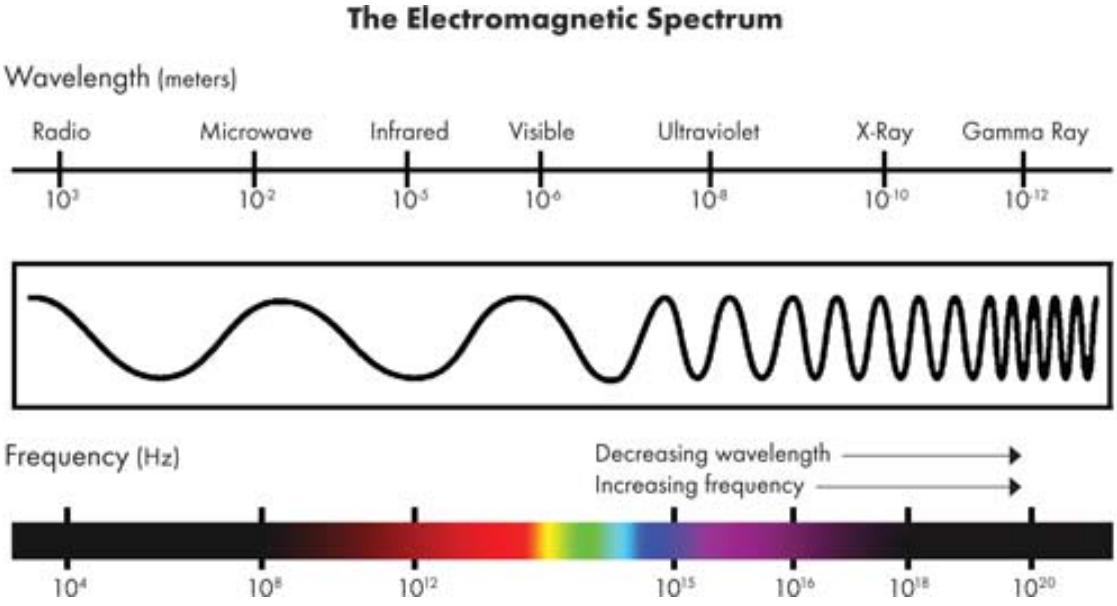

Figure 2: Frequency Spectrum Graph [6]

1.3 Knowledge and Skills required

This senior design project was a great way to build upon skills learned in the classroom. With regards to the hardware portion of our project, some skills that we drew upon included circuit design and simulation, hardware testing and

troubleshooting, digital signal processing, and communications theory including concepts of noise, bit error rate, and signal integrity.

In addition to Electrical Engineering skills, we used computer science and engineering skills in implementing the modulation and demodulation algorithms with Arduino.

2 Description of Solution

2.1 Our Solution

We live in a world that is in constant flux as engineers transport us into the future. In the early 2000's, the widespread success of the cell phone industry changed our society in ways no one ever would have thought was possible. The next paradigm shift is driver-less cars. If the past is any indication, it will happen sooner than we might think.

Autonomous cars currently use sensors and image processing cameras to ensure they stay in their lanes and don't get into accidents. However, all these sensors are internal to the vehicle itself. As such, our product seeks to provide a solution that works alongside these current solutions, but allows smart vehicles to establish a data link with other cars or city infrastructure to potentially create a mesh network of connectivity. This would make for a robust system - sensors and network connectivity - that would not only improve the safety of autonomous vehicles on the road, but could also provide the ability to optimize traffic routes of these vehicles as well. We decided to build this communication system such that it can use the LED headlights and taillights of cars to send visible light signals and create this data link. We are mainly focused on building the wireless communication channel as a proof-of-concept system for the viability of the technology. As such, the outcome of our project will be that we have a working data channel between an LED headlight and a light receiver, such that we can pass whatever data we want over this channel.

2.2 Project Objectives

Our project continued on the progress of last years project, Visible Light Communication (VLC), by Gemi Griffin, Alicia Chan, and Andrew Yu. In their project, they were able to use blinking LEDs and an array of photodiodes to communicate data at a maximum distance of 4 feet, with a data rate with less than 1 kilobit/second. Since we built a communication system tailored for vehicular communication, some of our main objectives were to improve on the distance, accuracy, and data rate of their project such that it would acceptable for the safety requirements of autonomous vehicles. In doing so, we aimed to show how VLC could be applied to automobile headlights to create an Internet-of-Things connected automobile network.

Our main objective from the start was to build a system that was robust enough to provide the basis for communication not only between vehicles and other vehicles, but also stop lights, street signs, and other city infrastructure, as illustrated in Figure 3. A vehicle to everything (V2X) network could be the future of optimizing everything in terms of safety and traffic on the roads, and we hope our project provides the foundation for such a system. In order for our

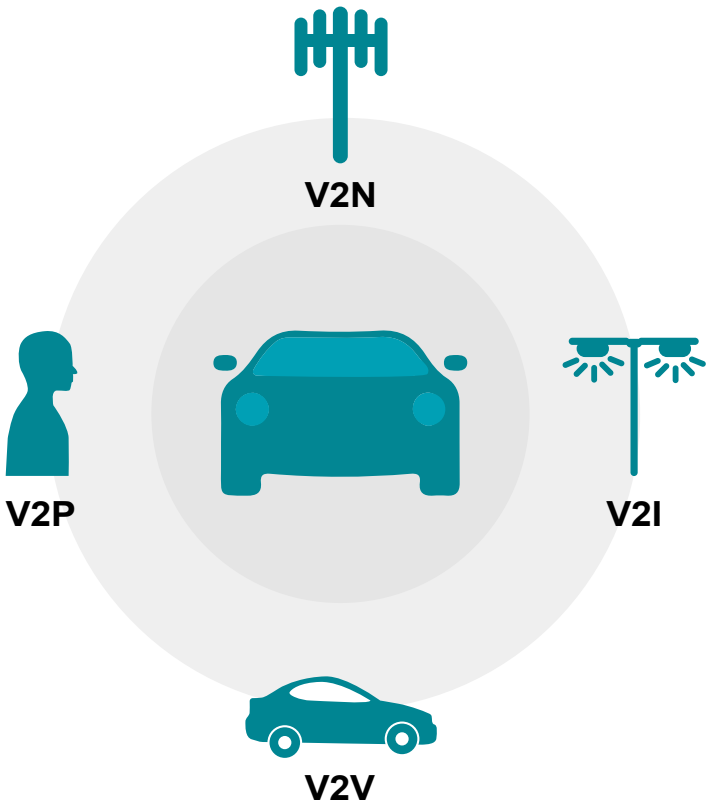

Figure 3: V2X Diagram [7]

system to be considered a success, we tested to ensure that our system worked a distance of tens of feet, had a very low bit error rate, and could pass data at a fast enough rate to ensure safety between autonomous vehicles.

3 Requirements

At the very beginning of our project, we laid out a complete list of the requirements that we wanted as essentially milestones for our project. We described these requirements in terms of functional requirements, or what the system must do, and non functional requirements, which are essentially what the system must be. These requirements provided us a guideline of what to aim for throughout the duration of our project, and helped keep us on track to designing a final product. Additionally, we operated under certain design constraints, which restricted us on how we should actually go about implementing our project.

3.1 Functional Requirements

- The system transmitter modulates visible light for data transmission
- The system receives visible light data, synchronizes with the data, and demodulates it

3.2 Non-functional Requirements

- The system provides a proof of concept for a Visible Light Communication system in vehicles
- The system is well-documented
- The system is fast enough to send visible light data over 10 kb/s
- The system provides reliable visible light communication at a range of over 20 feet
- The system has a bit error rate of less than 1 bit per 100 bits sent

3.3 Design Constraints

- The system is powered with voltage from the headlights of a car (12-14V)
- The system is built within a budget of $450
- The system uses VLC signals to send data

4 Design Alternatives Considered

4.1 Electromagnetic Waves

The first design decision that we had to make was how we wanted to go about passing the data for our communication system, which was ultimately the basis for our project. As mentioned in the problem statement, the radio spectrum is completely filled up in terms of frequency allocations, and so we wanted to look into forms of communication that use untouched or unlicensed frequencies of the electromagnetic spectrum. We ultimately had three choices: visible light communication, 5G, and Wi-fi. VLC and 5G use frequencies beyond the licensed part of the radio spectrum, while Wi-fi operates in the ISM band, which is a portion of the radio spectrum which is unlicensed and reserved for industrial, scientific, and medical use. All three also look like they could potentially be the future of vehicular communication.

We ultimately decided to go with visible light communication because there is already a framework in place for it to potentially be very useful, and very easy to implement. All new vehicles have LED headlights, and most street lights and street signs are moving in the direction of using LEDs as well, which provides the opportunity to create a system that not only communicates from vehicle to vehicle, but vehicle to everything. This possibility really interested us and drew us to using visible light communication for our system.

Another alternative that we considered, however, was making our system heterogeneous, in that it uses both visible light and 5G millimeter wave technology to pass data. We explored this alternative because we thought it could be interesting to show the safety of a heterogeneous system, as well as compare and contrast the connectivity of the two separate communication channels. However, we ultimately decided to only focus on visible light communication because we wanted to put all of our time and energy into making a single robust system.

4.2 VLC Modulation

After choosing to use VLC to pass data, we had to decide how we wanted to modulate the light for data transmission. Our group ultimately had two choices for the modulation scheme that we could use to transmit data. The first of these choices was standard On-Off Keying. This just means that the light being off would be decoded as a logic ’0’ while the light being on would be decoded as a logic ’1’. This modulation scheme is good in the sense that it is easy to implement. The problem with basic On-Off Keying, however, is that if multiple 0’s are sent in a row through the system, the light would appear off or look like its flickering.

Our other choice for VLC modulation was doing an OFDM modulation scheme (Orthogonal Frequency Division

Modulation). Whereas On-Off keying passes data by changing the amplitude of the light that it is passing, OFDM passes data through changes in the frequency that it is passing at. Even though this would solve the problem of the flickering lights, it would be much more complicated to build and test. Because of that, we settled on a system of On-Off Keying that uses a Manchester Encoding scheme to solve the problem of flickering lights. This encoding scheme is discussed in detail in section 7.1.

4.3 Microcontroller

For a microcontroller, we essentially had two options: an Arduino or a Raspberry Pi. Even though Raspberry Pis are are powerful and provide more functionality than Arduinos, Arduino provides and easy-to-use IDE and has all of the functions we would need to demodulate our data. We ultimately decided to use an Arduino as our microcontroller because it is much more user-friendly, while still offering us the proper amount of processing power for what we want it to do.

5 Project Design

5.1 Overview

The block diagrams help decompose our system into subsystems with inputs and outputs that help us better understand the functionality of the system as it relates to the actions of its separate parts.

5.1.1 Level 0 Diagram

The level 0 diagram provides a description of the system as it relates to its application context. For our system, it can be simply defined as a wireless communication system that passes data from object to object, specifically within the application of vehicles. Each transceiver is powered with 12-14 volts and can pass and receive data. This can be seen in Figure 4.

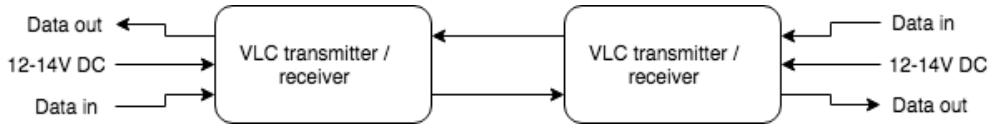

Figure 4: Level 0 Block Diagram

- Module: VLC transmitter / receiver
  - Inputs:
    - Data to be sent over the communication channels
    - 12-14 volts direct current
  - Outputs:
    - Demodulated data out
  - Functionality:
    - To modulate data that can be sent over flexible communication signals for use with vehicle data transmission

5.1.2 Level 1 Diagram

The level 1 diagram provides a visual idea of the system architecture, and the subsystems that will be implemented and how they relate to each to each other. For our prototype, we used an Arduino Uno as the microcontroller to control both of our circuits. This can be seen in Figure 5.

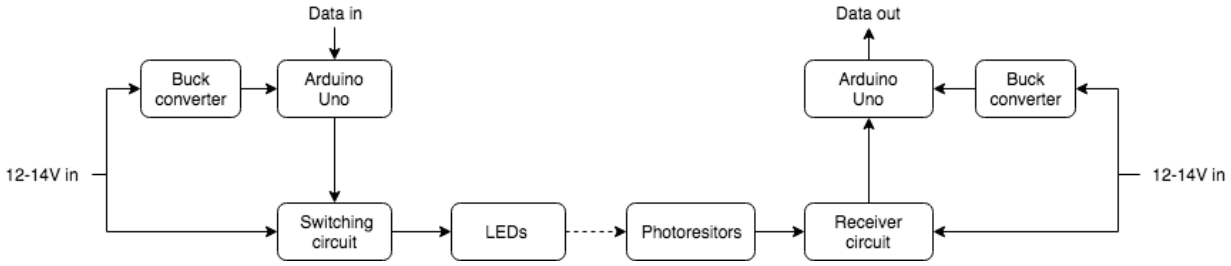

Figure 5: Level 1 Diagram

- Module: Switching Circuit / LEDs
  - Inputs:
    - 5 volt logic signal from Arduino
    - 12-14 volts direct current
  - Outputs:
    - Modulated light signal
  - Functionality:
    - To take a bit stream from the Arduino and modulate the LEDs based on that data

- Module: Receiver Circuit
  - Inputs:
    - Modulated light signal
    - 12-14 volts direct current
  - Outputs:
    - 5 volt logic signal to the Arduino
  - Functionality:
    - To receive a modulated light signal and covert it to its corresponding 5 volt logic signal for the microcontroller to demodulate

6 Hardware Implementation

6.1 Schematics

The schematics for the transmitter and the receiver of the VLC circuit are seen in the figures below. As seen in the transmitter schematic, we used a MOSFET hooked up to an Arduino to quickly turn the LED headlight on and off to transmit the data. Also, as seen in the receiver schematic, we used an array of photo-diodes to receive the light and then pass it to a floating threshold comparator to determine the threshold based on the intensity of the received light.

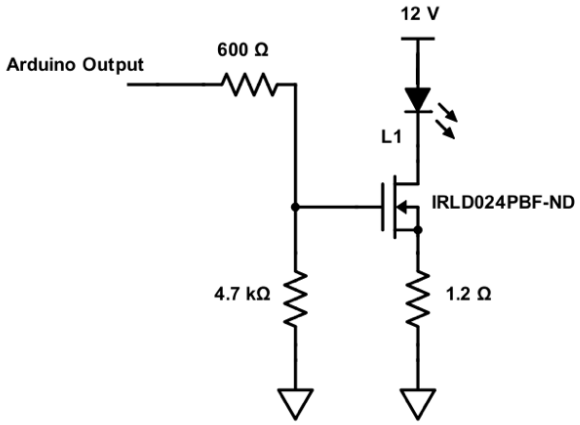

Figure 6: Transmitter Schematic for the VLC Circuit

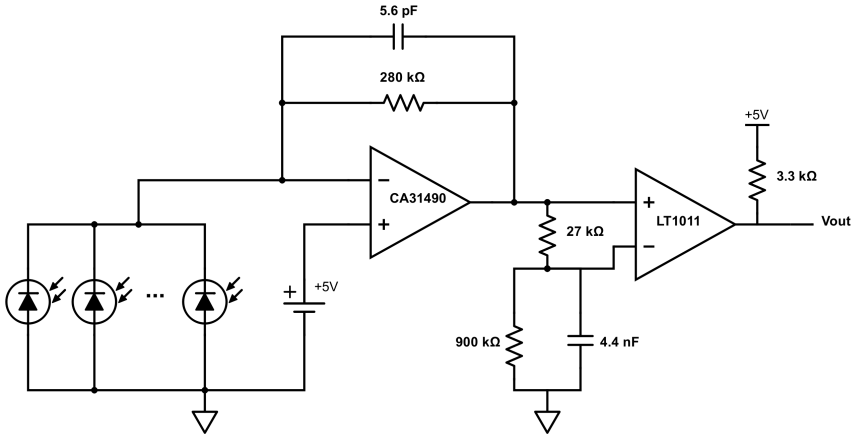

Figure 7: Receiver Schematic for the VLC Circuit

6.2 MOSFET

We chose to use a MOSFET in our design rather than having the Arduino directly connect to the headlight because it gives us more freedom in the voltage that we can use to power the headlights and in controlling the current that runs through the Arduino. By using the MOSET as a switch to turn on and off the LED, it takes control of the current and voltage and can operate as a separate module for debugging purposes.

6.3 Photodiodes

We decided on using photodiodes, rather than other light detectors such as phototransistors or photoresistors for several reasons. Photodiodes can generally operate faster than phototransistors, which is very important in a system like ours, where the light that the receiver is receiving is blinking at a very fast rate. This also gives us the ability to increase the data rate in the future if necessary. We also chose photodiodes over photoresistors because photodiodes are better for applications where the signal is either on or off, such as in digital communication.

6.4 Trans-impedance Amplifier

We decided on the CA31490 fast switching trans-impedance amplifier to convert the current signal from the photodiodes in a voltage. The amplifier has five volts to the non-inverting input because this voltage is required to achieve best performance from the photodiodes. The voltage and capacitor network acts as a low pass filter to remove a higher frequency signal that rides atop the 20 Khz square wave. At the output of the amplifier, the signal is a square wave with a low value of 5 and a high value of anywhere between 5 and 14 volts depending on how much light the photodiodes receive. The amplifier can be seen in Figure 8 below.

6.5 Comparator

The second half of our receiver circuit acts as a floating threshold comparator. To ensure our system works in various light conditions, we used a comparator. In this way, ambient lighting provides a DC bias, which can be filtered out and compared to our square wave. At one input is the original signal, say a square wave with a swing of 5-7 volts. At the other input is the DC bias of that square wave, or a straight line at 6 volts that is achieved by running the square wave through the low pass filter as shown in the figure below. At the output of our comparator is a signal that is either 5 volts or 0 volts, and we can input and demodulate this with the Arduino microprocessor. The comparator can be seen

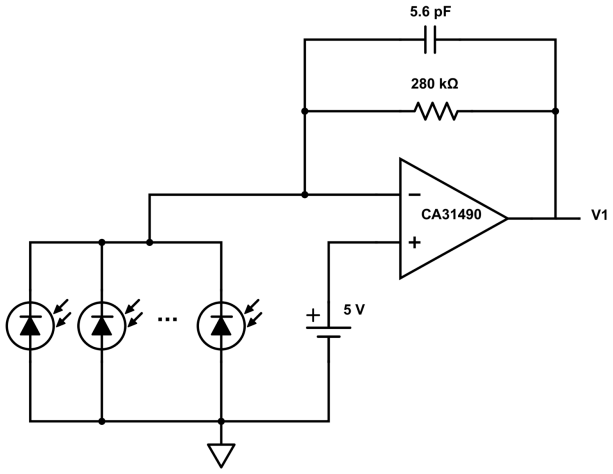

Figure 8: Transimpedance amplifier

in Figure 9.

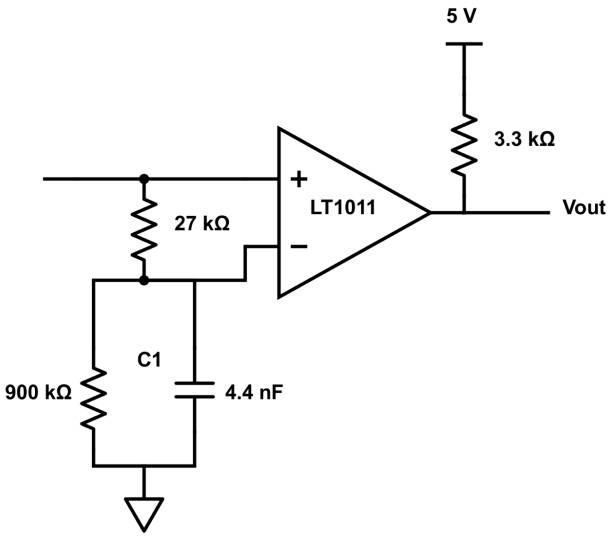

Figure 9: Comparator

7 Software Implementation

7.1 Manchester Encoding

7.1.1 Explanation

When you are passing data using visible light, a few concerns arise that are not problems with traditional communication systems. Since it is possible to see the data being passed, the brightness of the light can change depending on how many 1's or 0's you are passing at any point in time. For example, if the communication system were passing several 0's in a row, the light could turn off for a significant enough time to be noticeable to the human eye, and it would essentially look like the light was flickering.

Because of this idea, one of the most important concepts that we employed in communicating our data is Manchester Encoding. Manchester Encoding is a communication technique where instead of simply passing raw 1s and 0s through the communication system, the system instead passes a 1-0 for a 1 and a 0-1 for a 0 [5]. As shown in the example below, if our communication system was trying to pass the byte, 10100110, it would instead blink 1001100101101001 over the LED light, and then demodulate it on the receiving end in software. This is shown in Figure 10.

This ultimately allows us to control the duty cycle of the LEDs and ensure that it is always at 50%. With control over the duty cycle, we can ensure that the brightness of the LEDs stays constant no matter the data that is being passed. This also prevents the aforementioned scenario where it passes so many 0s in a row that you can see the light flickering.

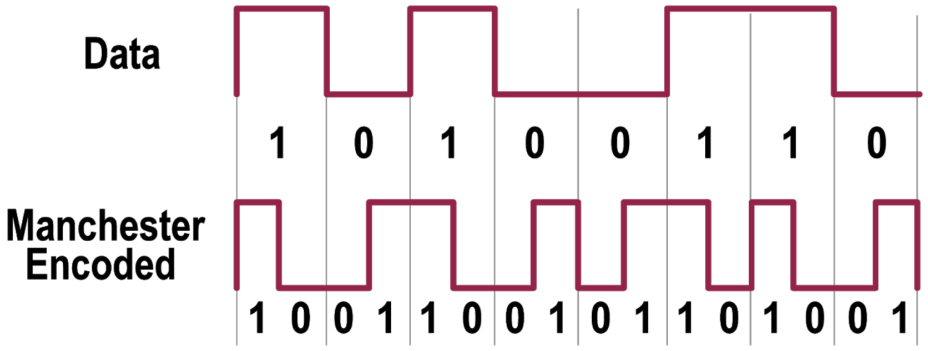

Figure 10: Manchester Encoding Example

7.1.2 Implementation

In order to include Manchester Encoding in our communication system, we decided to implement it on the software side, which is actually a design decision that we inherited from the Senior Design group last year. To implement this in software, we take the raw data that we want to pass, and do a bitwise comparison with a buffer of 0's that is double the size of the byte that we want to encode. We then read each bit of the raw data, starting with the most significant bit, and if it is a 1, we OR the buffer with logical 10, and if it is a 0, we OR the buffer with a logical 01. We then logical shift left the buffer twice then take the next bit of the raw data. The code for this can be seen in Appendix A1.

To Manchester decode the received data, we also do a bitwise comparison. For this comparison, we take a 16 bit Manchester Encoded sequence and convert it to its corresponding byte. To do so, we start with a buffer of eight 0's. From the Manchester Encoded bit stream, we check every other bit, and OR that bit value with the 8 bit buffer, logically shifting left after each comparison. This ultimately gives us the corresponding byte. The code for this can be seen in Appendix A2.

7.2 Demodulation

As explained in the previous section, the output of the hardware on our receiving end is a 5 volt logic signal. Based on that logic signal, it is the responsibility of the Arduino to sample from that signal and demodulate that data down to the exact bit stream that our transmitter sent. This is implemented in software, and the code for it can be seen in Appendix A4.

In order to demodulate, We start by sampling the signal at the fastest that the Arduino will let us, which is around 100kHz. This ultimately gives us several samples per bit, since we are sending data at 10-20 Kbps. Since Arduino only has a single processor, we cannot read and process the data simultaneously. To get around this issue, we read the samples into a buffer, as shown in Figure 11, and then process the data after reading. However, to make this work properly, we need to synchronize with the data as we demodulate it.

7.2.1 Synchronization Signal Processing

In order to properly demodulate the data, the Arduino needs to synchronize with the data signal so that it can down sample to the correct bit stream. In order to do this, we begin every data signal with a known synchronization signal. By doing it this way, our receiver is always checking for this synchronization signal, and once is receives it, can

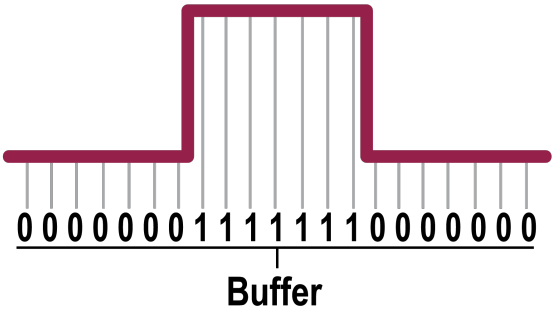

Figure 11: Buffer Sampling

determine the number of samples per bit based on this known signal. With this synchronization signal, our receiver can also find where the synchronization signal ends, and know exactly where to start reading the actual data from our bit stream. In synchronizing our data this way, it also makes our demodulating algorithm more robust in that it doesnt have to know the frequency of the received signal; it can demodulate signals at a range of different data rates.

7.2.2 Down Sampling

After processing our synchronization signal, our algorithm is able to calculate the average amount of samples that we are getting per bit. This is very important because it allows us to use that number to compare to the sample of the data buffer in order to down sample to the proper bit stream. Since we are using Manchester Encoding, we know that there will be no more than two 1's or 0's in a row. This makes our demodulation algorithm very easy, because we can compare our samples to the average number of samples per bit, and figure out which groupings of samples correspond to a 0, a 1, or two 0's or two 1's. From this calculation, we are able properly down sample our buffer. The implementation of this algorithm can be seen in Appendix A4.

8 Application of Communication System

8.1 Transmitter

In order to properly test our communication system, we needed a way to analyze its range and bit error rate. Since we did not have access to vehicles that have the framework to support our communication system, we instead decided to build a text messaging application that uses the VLC system to pass data. In doing so, we created a way to robustly test our communication channel, as well as demonstrate what happens at every step of our system.

The text messaging application starts by receiving input from the Arduino serial monitor. Using that input, it converts each character into its 8 bit ASCII equivalent, which it then Manchester encodes into its 16 bit equivalent, which is passed over our communication channel with the preceding synchronization signal. For example, if the system were sending the letter 'h', it would translate it into its ASCII equivalent of 104, which is 01101000 in binary, and ultimately 0110100110010101 when Manchester encoded. This can be seen in the figure below.

Figure 12: Data Steam Example

This Manchester encoded bit sequence is what is blinked through our LED in our communication system. The implementation of this can be seen in Appendix A3.

8.2 Receiver

After the Manchester encoded data is modulated through our LED, it is received through the hardware of our VLC receiver. Continuing with the example of passing the character 'h' over our system, the photodiodes in our receiver should create a current that corresponds to the amount of light that they are receiving, which can be seen in Figure 13 below.

Since the photodiodes in our system create a current that matches the amount of light that they are receiving, the output of our transimpedence amplifier is a voltage signal that follows the same form as the current that the photodiodes create,

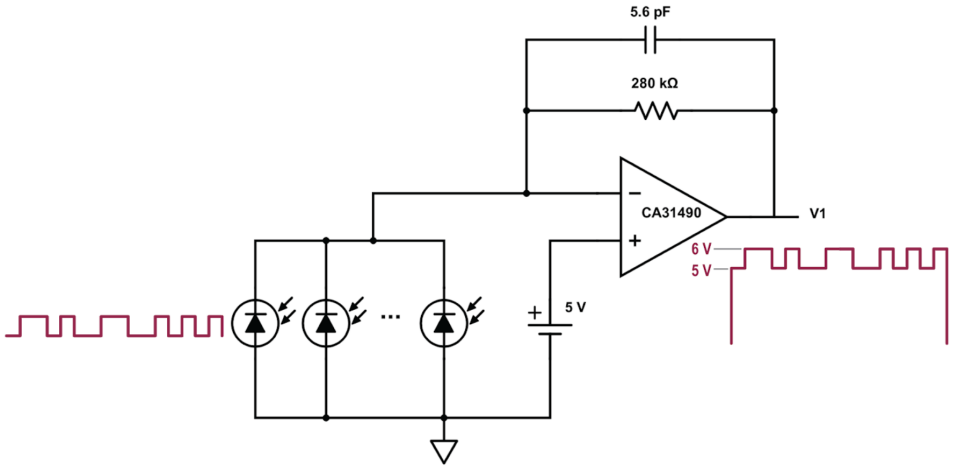

Figure 13: First Half of Receiver Demo

on top of a 5 volt reference. The upper limit of this voltage signal can be anywhere between 6 and 12 volts, depending on how bright our LED transmitter is and how close to the receiver it is. That voltage signal is then passed into our low pass filter, which creates a DC bias signal (seen in gray) that we can compare to with our comparator, as seen in the Figure 14. Our low pass filter, as explained in earlier section creates a DC bias signal, seen in gray, that can be compared to our logic signal. By using a DC bias comparator, we are able to demodulate signals no matter the ambient light conditions and no matter how bright our how close our VLC transmitter is. Ultimately, after comparing our signal with its DC bias, our hardware outputs a corresponding 5 volt logic signal, which is then processed and demodulated by our Arduino.

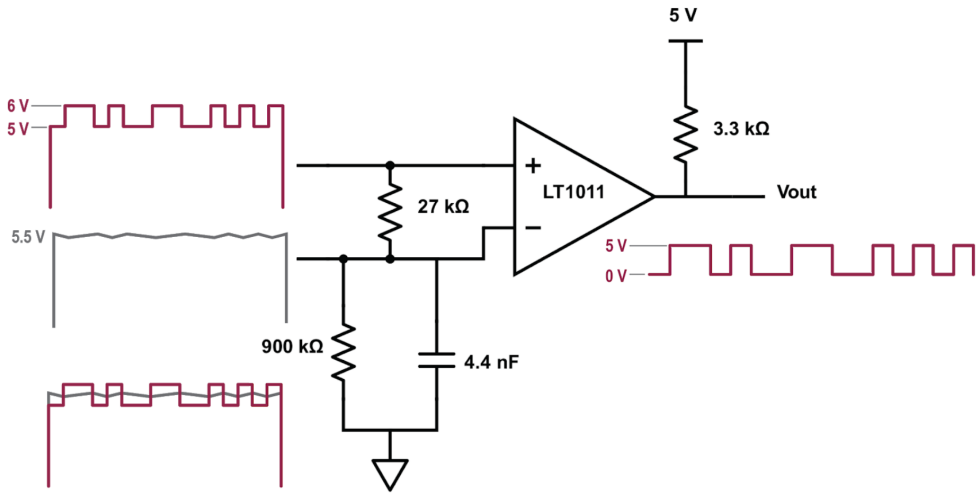

Figure 14: Second Half of Receiver Demo

9 Test Plan and Results

9.1 Test Plan

To test our circuit, we considered several factors to ensure the functionality of our system. In particular, we wanted to test signal integrity by testing the bit error rate in various light conditions and at various distances. For our distances and light conditions, we chose to test at 10, 20, and 30 feet both indoors and outdoors to replicate cars following at various distances such as stopped in traffic or moving on the freeway.

Considering this test plan we had a few quantitative benchmarks that we were trying to meet. For our data rate, we tested at 20 Kbps, because our contact at Qualcomm working on 5G V2V said that this is the industry standard rate for connected vehicle communication. Also, we tested to see if our system had a 10-3 or lower bit error rate per 100,000 bits to ensure an acceptable level of accuracy.

While our system would be used to transmit information such as speed, position, and heading between cars, we chose to test our system by implementing a text messaging application to make the data communication more interactive and user friendly. To do so, we decided to send a standardized test packet and compare it to the received message to determine the bit error rate. In particular, our standardized packet was 100 characters long and consisted of most of the ASCII characters. With 8 bits per character and 100 characters per packet, we sent 125 packets at each distance for a total of 100,000 bits in each test.

9.2 Indoor Test Results

Indoors, we tested in a classroom with the blinds open and ceiling lights on. Placing the transmitter at a distance of 10, 20, and 30 feet from the receiver, we sent the standardized test sequence of 100,000 bits at each distance. Next, we copied the received signal into a word document and used a program to scan the sequence of characters and compare it to the standardized sequence. After analyzing our results, we found that zero bit errors occurred at each distance. This was a great result because it demonstrates the reliability and accuracy of our communication system. A summary of these results in shown in figure 16 below, along with screen shots of the oscilloscope signal at each distance to demonstrate the integrity of our received square wave signal directly after the trans-impedance amplifier. It should be noted that the lower bound of our square wave was 5 volts, while the upper bound ranged from 10.6 volts at 10 feet to 7 volts at 30 feet. These results are promising because although peak to peak voltage decreases linearly with increased

distance, even at 30 feet there is still a 2 volt swing, plenty of headroom for the comparator to handle accurately. In fact, in earlier testing, we found that the comparator could accurately distinguish between the highs and lows of a signal with only several hundred millivolts peak to peak.

Distance	Bit Error Rate at 10 Kbps	Square Wave Swing at 20 Kbps
10 feet	0 errors/100,000 bits	5-10.6 volts
20 feet	0 errors/100,000 bits	5-9 volts
30 feet	0 errors/100,000 bits	5-7 volts

Figure 15: Indoor Test Results

9.3 Outdoor Test Results

Outdoors, we once again tested at 10, 20, and 30 feet, but this time in sunny daylight conditions. After sending the 100,000 bit test sequence at each distance and comparing it to the received sequence, we once again measured zero bit errors at each distance. However, outdoor testing provided a greater challenge, as the system required more calibration as well as lengthier and more accurate focusing of the headlight beam on the receiver. Daylight conditions negatively affected our system in another way, because we found that the bright ambient lighting interfered with our headlight signal and pushed the amplified signal above the rails of the trans-impedance amplifier, meaning that there was no way to distinguish a square wave within the hardware. However, we were able to solve this problem by implementing a daylight blocking filter, in our case a toilet paper roll. This simple fix provided a sort of sun shade that blocked out ambient lighting and allowed only the headlight to reach the photodiodes. After implementing we were able to send signals without error at 10, 20, and 30 feet as mentioned above.

9.4 Maximum Distance Testing

While we were limited by extension cord length and access to outlets outdoors, we were able to test for maximum distance indoors. After extensive testing, we found that 60 feet was the farthest distance across which we could reliably and consistently transmit datagrams. Past this distance, out receiver was still able to pick up messages, but the error rate was too high to ensure reliability. However, we are happy with this result because 60 feet is about how far apart cars would travel on the freeway.

10 Professional Issues and Constraints

10.1 Ethical Considerations

We have all been told many times that knowledge is power, and in almost no other profession is this as true as it is in engineering. Engineers are extremely responsible for their actions because their actions result in huge changes for the world at large. Their power, gained through specific knowledge, brings with it a tremendous responsibility to do a good job and contribute to positive change in the world; and not necessarily to make the most money. Engineers have to be very conscientious to not use their knowledge in a way that results in negative change for the world, and so must be very careful to create foolproof, safe products. The IEEE code of ethics seeks to foster this creative, honest, and virtuous community that actively pursues the common good.

Thinking about our project becoming widespread throughout the world forces me to think outside of the college perspective and get more insight about what it means to be a fully-fledged engineer. Our project could have enormous significance as the world moves rapidly towards automated driving vehicles. If done wrong, inter-vehicular communication could have tragic consequences for people all over the world. Because of this, engineers need to make sure to be diligent to be able to account for unexpected events to occur. They also need to be aware of all the ethical concerns of the products that they make and do their best to mitigate them.

Although our project was not derived from direct ethical considerations, it definitely carries with it some moral concerns because of the field that we are working in. When dealing with vehicular communication, these considerations are overwhelmingly grounded in the safety of the users. We are approaching a future where a large percentage of vehicles on the road are going to be autonomous, and so ethical considerations in this field will only become more important. In order to understand this fully, we need to outline the ethical considerations of our project in terms of justification for the project, what it means to be an ethical engineer, and the pitfalls and concerns that arise from our project.

The main purpose of our project is to create a new and innovative solution for vehicular communication that can be used in unison with current solutions, in order to provide new mediums of data transfer for cars. Although the purpose of the project does not have any direct ethical significance, the implications of how our project can be used to benefit the safety of the general public definitely carry some moral weight. This is because, as autonomous vehicles become more commonplace, their functionality affects the safety of everyone - drivers and passengers alike. Because of this, the need for reliable forms of communication will become even more crucial.

The key to dependable data communication is to use multiple channels to pass the data. Our project looks to do just this by providing an alternative medium to RF communication for vehicles. By providing multiple channels, one form of communication can take over when another form fails, exponentially increasing the reliability of passing the data. Thus, our project will hopefully benefit the safety of drivers by increasing the reliability of vehicular communication. With this, we are doing our part to improve our understanding of technology and avoid the possibility of technology injuring people, as are our duties outlined in the IEEE Code of Ethics.

10.2 Science, Technology, and Society

If spread throughout the world, our product would increase the safety of autonomous vehicles by providing another channel for automotive communication. It would affect anybody who owns an autonomous vehicle but would only affect them in positive ways by allowing for safer travel. The people using the product would likely not even know that the system is using our technology but would know that their car is capable of getting them where they need to go safely. Autonomous vehicles ultimately will lead to less vehicle-related deaths and less time spent in traffic which we see as both very positive outcomes. Our product satisfies a need of autonomous vehicles which will lead to a safer vehicle experience.

10.3 Civic Engagement

Our LED transmitter uses standard vehicle LED headlights to function and so it meets the standards set by the Department of Energy as well as the standards for brightness of cars on the road set by the National Highway Traffic Safety Administration under the Federal Motor Vehicle Safety Standard No. 108. Our LED's will also help save energy and coincide with the rapid increase of the use of LED's in modern lights which will help get the approval from the Department of Energy.

10.4 Economic

Our product consists of some raw electronic components, some op-amps, and a microcontroller. Because of this and the fact that it would be integrated into the existing headlights of automobiles, our product is very cheap to manufacture. Ideally, we would work with the car manufacturers to integrate our product design directly into the headlights for mass-production. We would be selling our product to the car manufacturers instead of just the common consumer. With the self-driving car revolution just on the horizon, we think our product will have a place in industry

for a very long time in many different automotive manufacturers. This results in creating a very steady income that will be available for years to come. If we were to mass-produce this product, we would be able to drive the cost of manufacturing down to around $30. This would allow us to price it cheaply and sell to a vast amount of customers. We believe that every car in the future should have a system like ours in place that will work even when all the RF bandwidth has been used. Because of this, we think our product is here to stay, and will provide much of the world with a safe driving experience.

10.5 Health and Safety

The safety of the passengers in the autonomous vehicles is our utmost goal in developing this product. The key to making our product safe is by using it in conjunction with other systems that are currently used. It will therefore supplement present forms of communication which will lead to a vastly safer and more reliable network that can work in many different conditions and keep the consumer safe.

10.6 Manufacturability

If we were to mass-produce our product, we could fit the entire system onto a circuit board which could easily be produced at any of the fabrication centers here in the Silicon Valley. Ideally, we would like to work with automotive companies to work this system into the present system of manufacturing that is already in place. It would be simple to install the chip when the headlights are installed and would not require many changes than what is already done.

10.7 Usability

The user will not have to interface with our device at all as it will all happen without them knowing. The vehicle manufacturer just has to install our product into the headlight and taillight of an autonomous vehicle for it to work.

10.8 Environmental Impact and Sustainability

Considering the fact that our Visible Light Communication project uses LEDs and incorporates existing hardware, our project is sustainable from an environmental perspective. One of the main advantages of VLC communication is that it can transmit information at very high data rates through on-off keying of a light source. This requires a light source with fast switching times, and because LEDs are semiconductor devices, they are able to accomplish

this while traditional incandescent lighting cannot. This is important because LEDs are much more eco-friendly than incandescent light, converting 95% energy into light and 5% to heat, thus requiring much less power to operate. For example, our project uses 36 Watt LED car headlights, which if converted to incandescent bulbs would require 84 Watts for the same brightness. Also, LEDs last up to six times longer than incandescent sources, resulting in fewer lights produced and less manufacturing materials depleted. When it is time for disposal of LEDs, this can be done in an environmentally friendly way because they contain no toxic materials like mercury, which is used in fluorescent sources. In the modern world, there is an increasing trend toward LEDs in applications such as car headlights, indoor lighting, street lights and camera flashes. Because VLC can be easily implemented with existing hardware, in our case LED headlights, it saves money and resources. Also, existing power sources could be utilized to power our components and microprocessors, so our project doesnt need any additional batteries or chargers. In fact, the hardware required for our project consists of only several resistors, capacitors, op amps, photodiodes, and microprocessors. While the signal conditioning is done by the hardware, the demodulation of our data is all done through software, which keeps costs down and requires fewer physical components. As a result of its simplicity, our product would be longer lasting and contribute an overall lower carbon footprint.

10.9 Lifelong Learning

Our project allowed us to gain a hands-on experience with circuit design and to see first-hand the effects of our design choices. We developed troubleshooting skills that will be extremely useful even outside the realm of engineering. We also learned a lot more about communication systems through this project and about what it takes to create a system that can reliably pass data. We hope that others in the future will be able to take our work in VLC and be able to distribute it in our society to solve the problem of limited RF spectrum. What we learned during this experience will certainly help us in the future to solve real-world problems and make the world a better place.

11 Conclusion

11.1 Development Timeline

The Gantt chart below shows the projected time line of our project that we defined at the beginning of Fall quarter. We planned on doing all of our research fall quarter, doing the design work for our project throughout winter quarter, and implementing our design towards the end of winter quarter and the beginning of spring quarter. For the most part, we were able to stay very on track with our projected timeline, however, we did deviate from it somewhat. In terms of testing, we planned to test our system throughout spring quarter, however, a delay in getting our demodulation algorithm to work pushed back our target date to start testing our full system. Furthermore, we planned on implementing error control into our system, however, we were overly optimistic about the time that we would have to implement error control and ultimately could not find the time to do so.

Figure 16: Timeline as Predicted

11.2 Performance Analysis

In the end, we were able to get a working system that can take in data on the transmitter side, encode this data, then send it via a beam of visible light to the other side. The receiver can accurately decode this data and show what was sent. Aside from basic functionality, some of our major accomplishments include:

- Communication at 20 Kbps

- Communication at up to 60 feet
- Communication indoors, at night, and in bright daytime conditions
- Zero bit errors up to 30 feet
- Scalable algorithm which includes synchronization, Manchester encoding, and active listening
- Low project costs

All of these results support our conclusion that a VLC system could be implemented cheaply and efficiently into automobiles to create a network of connected cars. While our project consisted of a text messaging system, an actual vehicular VLC system could consist of a standardized packet of around 30 bits, with the first 7 consisting of a synchronization code and the next 23 designated for speed, position, heading, and overhead. When a VLC line of sight connection is established, such as in car following scenarios or traffic, packets could continually be sent among automobiles to complement RF 5G technology.

11.3 Future Work

Having taken up a project from a previous senior design team, our main goals for our project were to increase the distance and data rate, decrease the bit error rate, and demonstrate the feasibility of VLC in an automotive/outdoor setting. While we are proud of our accomplishments, there is still much work to be done. Whether undertaken by future senior design teams or other individuals, there are several main features that require future work.

For example, given the time, we would have liked to implement a two way communication system. Future projects may include work in establishing not only two way communication, but looking at the creation of a network of vehicles, including protocols.

In addition, future work might look into solving the problem of how to deal with multiple signals. For example, how would a receiver distinguish between signals if it picks up light from several car headlights? Whether multiplexing by frequency or choosing based on signal strength, this problem poses an interesting challenge.

Also, it is important to address connectivity issues related to intra vehicular VLC communication. For example, our system tests were stationary, but future work may look into fine tuning the software and hardware such that data would still be able to be communicated even when the a car's headlight is moving around slightly or not completely focused on the receiver. Another angle might be experimenting with our system in harsh outdoor conditions such as rain or fog.

11.4 Lessons Learned

Ultimately, this project was a very rewarding experience. While it was a lot of work, it was fun to meet our original goals and see a project through from conception to completion. Senior design has been an invaluable part of our engineering education because it allowed us to build upon skills learned in the classroom. Whereas many ELEN classes are mathematical or analytical in nature, they provided a solid foundation and technical understanding that allowed us to build a product of our own. While this project provided many unforeseen challenges, a hands on experience such as this provided a great learning experience.

In addition to the technical skills learned in both hardware and software, we learned valuable lessons in teamwork and project management. For example, we learned how to set goals, choose designs, order parts, build and test, and troubleshoot setbacks. Also, we learned soft skills like networking with professors and professionals and presenting our project to a panel of judges at the end of the quarter.

In the end, the two biggest lessons we learned were to never be afraid to ask questions or reach out for help, and that the best way to learn is by doing, or building something hands-on.

References

[1] ”United States Frequency Allocation Chart,” United States Frequency Allocation Chart — NTIA. [Online]. Available: https://www.ntia.doc.gov/page/2011/united-states-frequency-allocation-chart. [Accessed: 15-Oct-2017].

[2] ”500 Megabits/Second with White LED Light” (Press release). Siemens. Jan 18, 2010.

[3] ”Introduction to OFDM - Orthogonal Frequency Division Multiplexing” (Press release). Keysight.

[4] H. Haas (2011, July). Wireless Data From Every Light Bulb [Video]. Available: http://www.ted.com/talks/harald_haas_wireless_data_from_every_light_bulb.html.

[5] H. Owano (2015, November 26) Using lights for communications? Haas, pureLiFi, see brighter future,Tech Xplore [Article]. Available: https://techxplore.com/news/2015-11-haaspurelifi-brighter-future.html.

[6] ”Electromagnetic Spectrum,” — Colourware. [Online]. Available: https://i1.wp.com/colourware.org/wp-content/uploads/2013/11/electromagnetic-spectrum.jpg. [Accessed: 2-Mar-2018].

[7] ”Cellular V2X” — Qualcomm. [Online]. Available: https://www.qualcomm.com/invention/5g/cellular-v2x. [Accessed: 26-Feb-2018].

[8] I. Jang, D. Kim, S. Lim, Y. Kim, and T. Kang. Method and Apparatus for Transmitting Data Using Visible Light Communication. US Patent 8634725B2, filed October 7, 2011, and issued January 21, 2016

Appendix A Arduino Code

A.1 ManchesterEncoding.ino

```
void setup() {
  Serial.begin(9600);
  delay(2000);

  byte value = 0b00000001;
  unsigned int encoded = 0;
  for (byte b = 0; b < 8; b++) {
    encoded <<= 2;

  if (value & 0b10000000) {
      encoded |= 0b10;
    } else {
      encoded |= 0b01;
    }
    value <<= 1;

    }
  Serial.println(encoded, BIN);

  for(int i = 15 ; i >=0 ; i--)
  {
    Serial.print(bitRead(encoded, i));
  }
}

void loop() {}
```

A.2 ManchesterDecoding.ino

```
int incomingByte = 0; // for incoming serial data

void setup() {
```

```
Serial.begin(9600);

    //Test Values
    int test[16] = {0,1,1,0,1,0,0,1,1,0,0,1,0,1,0,1};
    int test1[16] = { 0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,0};
    int test2[16] = {0,1,1,0,1,0,0,1,1,0,1,0,1,0,1,0};
    int test3[16] = {0,1,0,1,0,1,0,1,1,0,0,1,1,0,0,1};

    manchester_decode(test1);
    manchester_decode(test2);
    manchester_decode(test3);

}
void loop(){}

void manchester_decode(int arr[16])
{
    incomingByte = 0b0;
    for(int i = 0; i < 16 ; i++)
    {
      incomingByte <<= 1;
      incomingByte = incomingByte | arr[i];
    }

    unsigned int decoded = 0;
    for (byte b = 0; b<8; b++) {
      decoded <<= 1;
      if ((incomingByte & 0b1000000000000000)){
        decoded |= 0b1;
      } else{
        decoded |= 0b0;
      }
      incomingByte <<= 2;
```

```
  }
  Serial.println(decoded, BIN);
  Serial.println((char) decoded);
}
```

A.3 Transmitter.ino

```
#define PERIOD 46

// Transmit 3 bytes at a time
int8_t incomingByte[3];
uint8_t i = 0;

void setup() {
    // opens serial port, sets data rate to 9600 bps
    Serial.begin(9600);

    // MOSFET gate connected to pin 13 of the Arduino
    pinMode(13, OUTPUT);
    digitalWrite(13, HIGH);
    delay(1000);
}

void loop() {
       // send data only when you receive data:
       if (Serial.available() > 0) {

              // read the data to send:
              incomingByte[i] = Serial.read();

              // Fill the rest of the buffer with white space if it's not completely full
               if(i == 0 && (char) incomingByte[i] == '\n')
               {
                  incomingByte[0] = ' ';
                  incomingByte[1] = ' ';
```

```
                incomingByte[2] = '\n';
                i+=2;
              }

              if(i == 1 && (char) incomingByte[i] == '\n')
              {
                incomingByte[1] = ' ';
                incomingByte[2] = '\n';
                i++;
              }

            i++;
      }

      // Transmit every 3 bytes it reads
      if(i == 3)
      {
        transmit();
        i = 0;
        digitalWrite(13, HIGH);
        delay(3);
      }
}

void transmit() {

  //Start By sending synchronization signal
  sendSynchSignal();

  digitalWrite(13, LOW);
  delayMicroseconds(PERIOD);

  // Transmit data
  for(int i = 0; i < 3; i++)
  {
```

```
    for (int j = 7 ; j >=0 ; j--)
    {
      if (bitRead(incomingByte[i], j) == 1)
      {
        digitalWrite(13, HIGH);
        delayMicroseconds(PERIOD);
        digitalWrite(13, LOW);
        delayMicroseconds(PERIOD);
      }
      else{
        digitalWrite(13, LOW);
        delayMicroseconds(PERIOD);
        digitalWrite(13, HIGH);
        delayMicroseconds(PERIOD);
      }
    }
  }
}

void sendSynchSignal(){
     digitalWrite(13, LOW);
     delayMicroseconds(PERIOD);

     digitalWrite(13, HIGH);
     delayMicroseconds(PERIOD);

     digitalWrite(13, LOW);
     delayMicroseconds(PERIOD);

     digitalWrite(13, HIGH);
     delayMicroseconds(PERIOD);

     digitalWrite(13, LOW);
     delayMicroseconds(PERIOD);
```

```
    digitalWrite(13, HIGH);
    delayMicroseconds(PERIOD);

    digitalWrite(13, LOW);
    delayMicroseconds(PERIOD);

    digitalWrite(13, HIGH);
    delay(3); //wait while processing

}
```

A.4 Receiver.ino

```
int digitalPin = 0;
int x = 0;

// Buffer size for the synchronization signal samples
#define BUFFER_SIZE 100

// Buffer size for the data signal samples
#define DATA_BUFFER 500

volatile int synchSamples[BUFFER_SIZE];
volatile int dataSamples[DATA_BUFFER];

// Read data in 3 byte packages (limited by Arduino's single threadedness and low memory for
    variables)
int byte_one[16];
int byte_two[16];
int byte_three[16];

// The average number of samples per period based on the known synchronization signal
int avgSampleWindow;

volatile bool startTrigger = false;
```

```
int i;

void setup()
{
    // Startup delay
    delay(3000);
    Serial.begin(115200);
}

void loop()
{
    i = 0;
    while(1)
    {
    x = digitalRead(digitalPin);

    // Starts reading when it receives the first 0
    if(x == 0 )
    {
             startTrigger = true;
    }
        if (startTrigger)
    {
             synchSamples[i] = x;
             i++;
        }

    if(i == BUFFER_SIZE)
    {
             startTrigger = false;
              break;
    }
    }
  synchProcessor();
```

```
  i = 0 ;

  while(1){
     x = digitalRead(digitalPin);
   if(x == 0 )
      {
        startTrigger = true;
      }
  if (startTrigger)
      {
        dataSamples[i] = x;
        i++;
        //Serial.println(i);
      }

      if(i == DATA_BUFFER)
      {
        startTrigger = false;
        dataProcessor();
        break;
      }
  }
}

void synchProcessor(){

// Uncomment for Debugging
//  for (int j = 0; j < BUFFER_SIZE ; j++)
//  {
//    Serial.print(synchSamples[j]);
//  }
//  Serial.println("");

  for (int k = BUFFER_SIZE - 1; k >= 0 ; k--)
```

```
  {
    if(synchSamples[k] == 0 && synchSamples[k-1] == 0)
    {
      //Calculate the Average Sample Window From Synchronization Signal
      avgSampleWindow = (int) round((float)(k+1)/ 7.0);

// Uncomment for Debugging
//     Serial.print("synchSamples in Buffer: ");
//     Serial.println(k+1);
//     Serial.print("Average Window Size: ");
//     Serial.println(avgSampleWindow);
      break;
    }
  }

// Serial.println("");
  i = 0;
}

void dataProcessor(){

// Uncomment for Debugging
// for (int j = 0; j < DATA_BUFFER; j++)
//   {
//     Serial.print(dataSamples[j]);
//   }
//   Serial.println("");

   for (int j = 1; j < DATA_BUFFER - 1; j++)
    {
      if(dataSamples[j] != dataSamples[j-1] && dataSamples[j] != dataSamples[j+1])
      {
        dataSamples[j] = dataSamples[j+1];
      }
    }
```

```
// Uncomment for Debugging
//   for (int j = 0; j < DATA_BUFFER; j++)
//   {
//     Serial.print(dataSamples[j]);
//   }
//   Serial.println("");

  int bits [49];

  int k = 0;
  int currentBit = dataSamples[0];
  int bitCount = 1;

  int z = 0;
  for( z ; z < 49; z++)
  {
    if(k == DATA_BUFFER)
      break;

    bits[z] = dataSamples[k];
    while(dataSamples[k] == currentBit)
    {
      k++;
      bitCount++;
    }

    if(bitCount > 1.5*avgSampleWindow)
    {

      bits[z + 1] = currentBit;
      z++;
    }
    bitCount = 1;
    currentBit = dataSamples[k];
```

```
  }

// Uncomment for Debugging
// for(int z = 1; z < 49; z++)
// {
//   Serial.print(bits[z]);
// }
// Serial.println("");

  for(int z = 0; z < 16; z++)
  {
    byte_one[z] = bits[z+1];
    byte_two[z] = bits[z+17];
    byte_three[z] = bits[z+33];
  }
  manchester_decode(byte_one);
  manchester_decode(byte_two);
  manchester_decode(byte_three);
  i = 0;
}

void manchester_decode(int arr[16])
{
    int incomingByte = 0b0;
    for(int i = 0; i < 16 ; i++)
    {
      incomingByte <<= 1;
      incomingByte = incomingByte | arr[i];
    }

    unsigned int decoded = 0;
    for (byte b = 0; b<8; b++) {
      decoded <<= 1;
      if ((incomingByte & 0b1000000000000000)){
        decoded |= 0b1;
```

```
      } else{
        decoded |= 0b0;
      }
      incomingByte <<= 2;
  }
  Serial.print((char) decoded);

}
```