Visible Light Communication Technical Report

Jayanth Shenoy
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas, USA
jayanth.shenoy@utexas.edu

Abstract – This paper fully documents the process of Visible Light Communication (VLC) development on the Arduino. The goal of the project was to build an embedded communication system that can transmit and receive basic messages encoded in various pulses of light. Overall, the entire development process could be broken down into three critical stages: 1. Hardware Setup, 2. Communication Algorithm, and 3. Software Implementation. Each stage had its own unique set of challenges that mostly stemmed from the fact that VLC in and of itself is asynchronous. In addition to establishing precise timing, other important features of VLC include the proper focus of light and the need to filter out environmental light noise.

I. Introduction

Given the world's constantly growing desire to collect and analyze data, development of reliable, secure, and fast methods of communication are becoming more important. The phenomenon of Visible Light Communication is a powerful way to manipulate and move data. Transmitting data through visible light pulses results in both a high-speed and highly secure means of communicating information. Additionally, the low-cost of VLC and the possible integration of VLC devices into everyday lighting serve as further motivation for the project.

The Visible Light Communication embedded device for this project involved a single transmitter sending data to a single receiver. Ideally, the VLC system would be able to encrypt bit data into LED pulses and send them at a speed so fast that the human eye would not be able to perceive the flashing of the LED light. Developing this VLC device involved finding and interfacing embedded electronics for a proper hardware setup, creating proper communication transmission algorithms for VLC, and implementing the algorithms on the embedded system for practical use.

II. Hardware Setup

A. *Using Arduino as the Proper Microcontroller*

Previous research into VLC conducted by Disney Zurich, showed that the Arduino, given its simple structure, had enough power to drive the Visible Light Communication process. As a result, the Arduino Uno board was used as both the transmitter and receiver microcontroller for the project. To verify this thought, several tests were done with the Arduino to analyze its ability to transmit and sample light.

Using a basic script that toggled the light on the transmitter side and printed analog read details on the receiver side, it was possible to determine how fast the Arduino could sample light waves. The setup of the transmitter and receiver are detailed in the diagrams below.

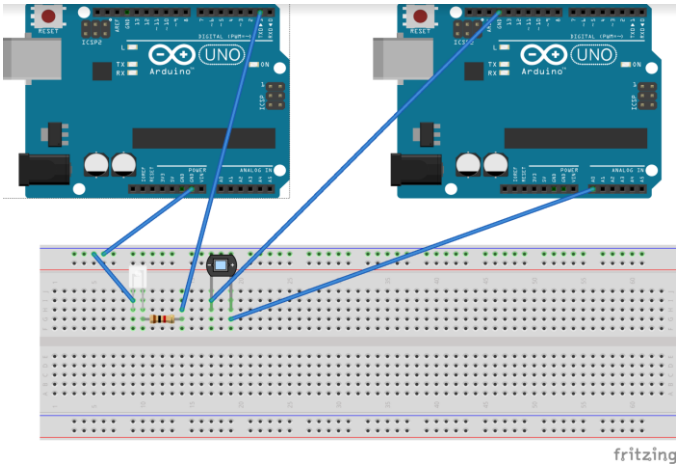

Fig. 1. Arduino fritz with VLC transmitter on the left and receiver on the right

At a first glance, it did not seem that the Arduino was feasible given the slow sampling rate of the default Arduino settings. However, several changes were made to the software. The first change involved a reset of the ADC prescaler on Arduino. The default prescaler value for the ADC was 128, meaning that the max sampling rate of the Arduino ADC was on the order of milliseconds. However, by changing the default prescaler value, the Arduino could sample data from the ADC at a rate in the order of microseconds, which is much more suitable for the high data rates of VLC. The other change that was made involved switching from a delay based busy-wait system to an interrupt driven system. Based on several datasheets and specs for the Arduino, the delay function in Arduino is known to be inaccurate and can cause timing issues while the data rates are high. After making these changes, it was straightforward to conclude that the Arduino was fast enough for VLC.

B. *AC Lightbulb Transmitter*

In order to integrate VLC with everyday lighting, it was important that consumer LED lightbulbs were tested. Most of these consumer LEDs are 60 watt AC lightbulbs that cannot be directly connected to a DC power source such as an Arduino. Therefore, a relay was needed in order to build a proper transmitter circuit involving an LED.

To build the AC VLC circuit, a relay was placed along a lightbulb socket wire to facilitate the conversion of AC to DC. Compared to opening the bulb to access the prebuilt relay

inside, this method was more efficient. The first type of relay tried was a contact relay, which mechanically converted power at the zero crossings. Because of the mechanical nature of the contact relay, there was significant, noticeable lag in the light bulb flashes compared to the DC counterpart. As a result, a solid-state relay was used for the final AC lightbulb circuit built. The circuit diagram of this design is below

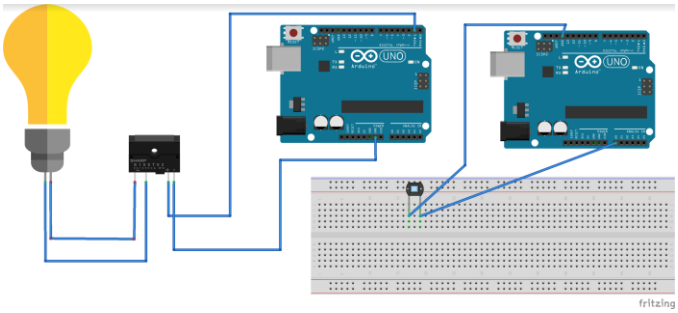

Fig. 2. Arduino fritz with AC light VLC transmitter on the left and receiver on the right

Although the AC lightbulb worked as a viable option when the symbol period was high, at higher data rates, the transmitter was unable to properly send data due to the flicker phenomenon. When the symbol rate was set to a very low period (30 microseconds for example), the lightbulb flashes were not perceptible to the human eye, but the lightbulb itself would occasionally flicker. The flicker from the AC lightbulb can cause major issues for VLC as the receiver could mistake a flicker for a very long low signal.

Because AC light itself is constantly modulating (even when it is set to high), modulating AC light for VLC created many noticeable timing issues. When sampling data from AC light at highspeed, the samples would modulate not because the Arduino was modulating them, but because the alternating current itself was modulating, making it difficult to detect what was actually a light flash versus what was just part of the natural AC light intensity variation. In addition to the issues of modulating and already sinusoidal light waveform, extra time to process the AC to DC conversion in the relay could have been a second cause of the flicker. As a result, a light bulb flash for AC light would need to flash at a 50-microsecond symbol period to not be perceptible to the human eye while a DC lightbulb would only need to flash at a 2200 microsecond symbol period to not be perceptible. Thus, it was decided that a strong DC light was more efficient for VLC since it could technically send more data at a slower speed than AC light.

C. DC Lightbulb Transmitter

The final setup developed for the VLC project was based on a DC bulb that operated at 12 volts. As a result, this DC bulb was brighter than the standard 5-volt DC LED and easier to modulate than the 60-Watt AC lightbulb. The symbol period required for the human eye to not perceive the light flashes of this bulb was 900 microseconds.

Unlike the AC circuit, a relay was not needed for the Arduino to control the light; however, because the DC lightbulb has a load resistance, so the current from the Arduino alone was not high enough to power the lightbulb. To solve this issue, a p-type MOSFET transistor was used to control the flow of current to the 12-volt bulb.

Additionally, the 12-volt bulb also required a 12-volt power source since the maximum amount of voltage that an Arduino can provide is 5-volts. A variety of 12-volt batteries could be used or lower-voltage batteries could have been placed in series. The circuit diagram of this design is below

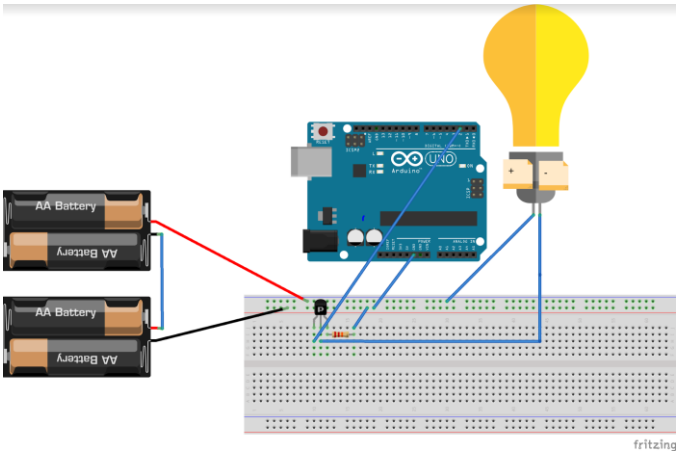

Fig. 3. Arduino fritz with 12 volt DC light VLC transmitter connected with MOSFET

D. Receiver Photodiodes

Photodiodes from the beginning were a clear choice for analog reads on the receiver end. They have the ability to pick up constant changes in light intensities at much higher rates compared to other similar devices, such as solar panels. Today, photodiodes are used in many optical technologies because of their ability to analog read at high data rates. A variety of photodiodes were tested on the receiver end.

In many initial tests conducted, the proper focus of the photodiode on the light source was critical to the receiver's analog read of light samples. Whenever the photodiode was misaligned, the serial monitor of the receiver would not be able to detect any light signal changes of the transmitter.

Moreover, although different photodiodes produced different absolute voltage readings on the receiver serial monitor, the differences between high and low light values remained constant despite the change in photodiodes. As expected, the PIN photodiodes that were tested produced less varied voltage readings compared to the Avalanche photodiodes. For the application of VLC, the PIN photodiodes were preferred because the high sensitivity of Avalanche photodiodes would make the receiver more vulnerable to ambient light interference.

III. Communication Algorithms

For Visible Light Communication, the algorithm for encoding and decoding bits is crucial for successful

transmission of messages. The three main algorithms that were considered were binary encoding, frequency encoding, and Manchester encoding. Each of these individual algorithms has its own advantages and disadvantages for VLC, but for the purposes of this project, Manchester encoding made the most sense to use.

A. Binary Encoding

As suggested by its name, this basic algorithm encodes high bits as on pulses and low bits as off pulses. The primary advantage to this algorithm is its low implementation complexity. However, at very high data rates (in the order of milliseconds), this algorithm causes the VLC device to fall out of synchronization. As a result, the binary encoding algorithm only works in the order of seconds. The algorithm would also have issues when the transmitter sends a long series of low bits. The light pulses would be low for a very long time at a high data rate, allowing the human-eye to perceive flicker in the light transmission.

In addition, the algorithm depends heavily on a set threshold value manually calibrated at the start of transmission. Depending on the amount of ambient light, the VLC threshold that determines whether a voltage received is low or high can vary dramatically, causing unreliable readings in certain cases.

B. Frequency Encoding

In theory, frequency encoding is the most powerful algorithm that has the greatest ability to counteract interference from ambient light. By operating a VLC device that only decodes bits at high frequencies, ambient light waves operating at much lower frequencies can easily be filtered out. Frequency encoding would also not have the long series of zeros problem of binary encoding and would be much easier to synchronize at high data rates. Many infrared based communication systems such as remote controls use a frequency encoding schema.

On the other hand, the disadvantages of frequency encoding for VLC lie in its application on the Arduino. The Arduino's processor and clock are simply not fast enough to handle the extremely high light toggle rates required for frequency encoding. Moreover, the implementation complexity of frequency encoding is higher than that of the other two algorithms discussed.

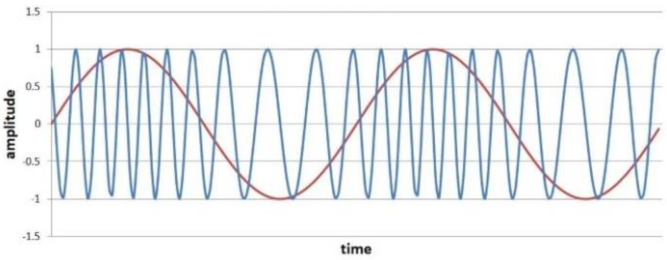

Fig. 4. Example of frequency encoding of a continuous signal where the values of 1 and -1 are encoded

C. Manchester Encoding and Preamble

Manchester encoding is essentially an implementation complexity versus accuracy tradeoff between binary encoding and frequency encoding. In this algorithm, high bits are encoded in transitions from low to high light intensity (rising edges) and low bits are encoded in transitions from high to low light intensity (falling edges).

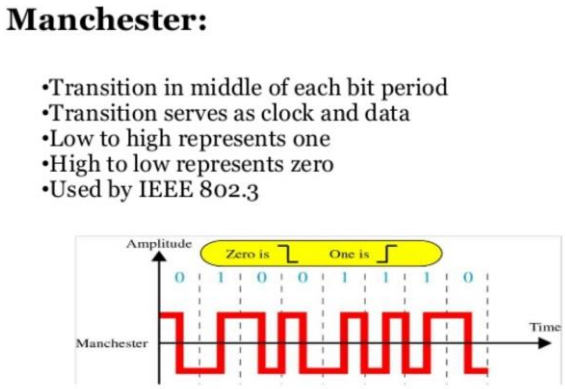

Fig. 5. Diagram of Manchester encoding

The main advantage of Manchester encoding is that the light source will be perceived as "on" by the human eye because the bulb itself will never stay low for a long period of time due to the encoding in edges. The act of encoding bits in rising edges and decoding bits in falling edges is also relatively simple computationally, resulting in less computational time lag between transmitter and receiver.

Taking advantage of this unique edge-based feature of Manchester encoding, a preamble sequence consisting of six high half bits was developed to synchronize the VLC device in this project. At the beginning of each repetitive message transmission frame, the preamble sequence sends 6 high half bits followed by a message starting with zero and ending with one. This specific bit pattern will never show up in transmission of the actual message because 6 consecutive high half bits is not part of the Manchester encoding scheme for any sequence of actual message bits. Having this specific preamble allows the receiver to synchronize itself to the transmitter. The receiver begins decoding a sequence of bits only if it receives the preamble sequence. After receiving the preamble, the receiver proceeds on to decode the next known fixed amount of half bits as part of the message sequence.

IV. Software Implementation

The software for VLC was written in the Robot C language for Arduino. Naturally, there are two programs to run VLC, one for the transmitter and one for the receiver. The following provides an overview of the individual files related to the software:

1) VLC Transmitter

a. VLCtestModuleTransmitter.ino – Arduino main loop file that contains periodic interrupts for sending data
b. FIFO.cpp (.h) – Code for circular buffer used to store encoded message bits
c. VLCtransmitter.cpp (.h) – Manchester encodes original message data and stores encoded bits into FIFO

2) VLC Receiver
a. VLCtestModuleReceiver.ino – Arduino main loop file that contains periodic interrupts for receiving data
b. VLCreceiver.cpp (.h) – Decodes message data and prints messages to serial monitor

A. Transmitter Data Flow

A message is sent via VLC by first modifying the message variable code in the VLCtestModuleTransmitter.ino. The user must specify the exact message and the length of the message in bits. During the setup function phase, the VLC transmitter processes the message bits into half bits based on Manchester encoding. The size of the FIFO will be exactly twice the length of the message since the FIFO will store half bits. The user must also specify the symbol period of each half bit in microseconds at the top of this file.

Once the FIFO is filled with the message, the periodic interrupt function in VLCtestModuleTransmitter.ino interrupts at every half bit and the corresponding digital write is called at the DC lightbulb port. The digital write has been optimized and the direct PORTD (GPIO 2) bit value is changed rather than calling the default Arduino digital write function.

B. Receiver Data Flow

The receiver begins by capturing half bits from the photodiode connected to a PWM analog port. The user must specify the length of the message in bits and the half symbol period in microseconds. An interrupt in the VLCtestModuleReceiver.ino called periodically every half bit, which should be the same value as in transmitter, samples the ADC looking for the preamble value. Once the preamble is found, the interrupt captures various message samples and calculates the difference to decipher the edges. The respective edges are then decoded into bits and attached to a message string. This process repeats after the length of the string has reached the number of bits specified by the user. The received message is printed to the serial monitor.

V. Conclusion and Future Work

A. Current VLC device and Its Constraints

The final VLC prototype using the 12-volt DC LED with Manchester encoding communicates VLC successfully to a certain degree. The maximum data rate of this VLC is approximately 1100 microseconds. Ideal environmental conditions for VLC occur when there is little to no ambient light present. By properly adjusting the threshold value in the receiver, the VLC device will successfully transmit data when moderate ambient light is present. Occasionally, at the the start of transmission, several seconds of inaccurate messages are returned by the receiver until the device can successfully resynchronize.

B. Enhancements to VLC

The next steps for the VLC device are to develop more error checking mechanisms. For example, in the receiver, after receiving the same message a certain number of times, the receiver should declare that message as the “actual” message and stop receiving. Although this process may be naïve, it will provide the receiver with a bit more inside on the accuracy of the message being received.

Once the VLC is more reliable at high data rates, VLC based application features can be added to the devices. These include generation of truly random bit passwords on the transmitter end and Android app connectivity display on the receiver end.

In general, finding different ways to optimize the VLC hardware and software can reduce the time lag between receiver and transmitter. By optimizing timing, the accuracy of VLC message transmission will become more reliable. Other possible changes to the VLC could be development of custom relay hardware for transmission on AC lightbulbs.

In addition to optimizing timing, the VLC device can be optimized based on distance between transmitter and receiver. A hardware based solution to this problem would be to develop a focus for the DC light source on the transmitter.