Light-Based Communication

Final Report

*Word Count: **1538** Words*

Wireless Sensor Networks

Prof. Petros Spachos

T.A. Madison McCarthy

Group 1: Ian Cabral, Jordan MacIntyre

December 2, 2016

1. Introduction

1.1. Background

While LEDs on their own provide for a great source of ambient lighting, they also serve as an excellent source for wireless communication and the transmission of data. On their own, LED lights offer many advantageous properties, including high brightness, reliability, low power consumption, and a long lifetime. However, by simply modulating the intensity of an LED, one can send data wirelessly to multiple receiver devices at large data rates - all while remaining unnoticeable to the human eye.

Through transmitting data via light, synchronization between devices is made easy, and data can be received by various devices all at the same time. In turn, this results in a significantly smaller energy consumption compared to already in-place methods for wireless communication. Additionally, a stronger sense of wireless security can be realized, as only devices within a direct line-of-sight to the transmitted light can receive information. Lastly, the transmission of data through light proves harmless to the human body, and can be seen as an added benefit towards preventing any health-risks associated with current wireless communication technology.

A few applications made possible through light-based communication include underwater communication systems, vehicle to vehicle messaging, and indoor broadcasting systems. Figure 1 below illustrates how introducing light communication into vehicles would allow for advanced knowledge of actively breaking vehicles ahead, however this could be easily expanded into transmitting any traffic delays or car accidents up-ahead. Similarly, Figure 2 illustrates how light could be used to transmit data to several devices, all at the same time.

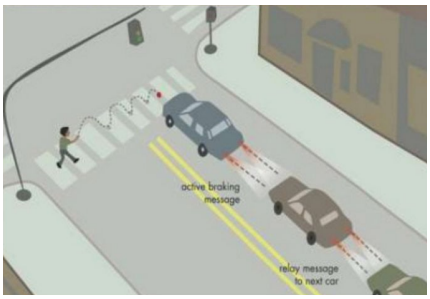

Figure 1: Vehicle to Vehicle Messaging

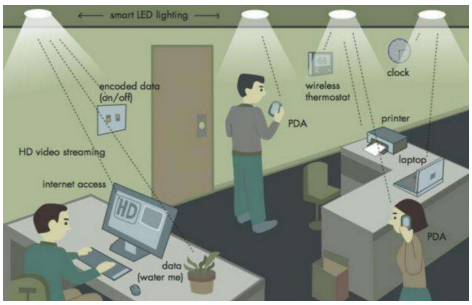

Figure 2: Indoor Broadcasting System

2. Implementation

2.1. **Problem Overview**

The system design was carefully developed using abstractions, so that a system model could be developed which highlighted the important elements necessary for the successful completion of the project. The project was broken into 4 major components: the transmitter, the receiver, the controller, and the viewer. Each of these components were created using a mix of hardware and software. The abstract system model can be seen in Figure 3.

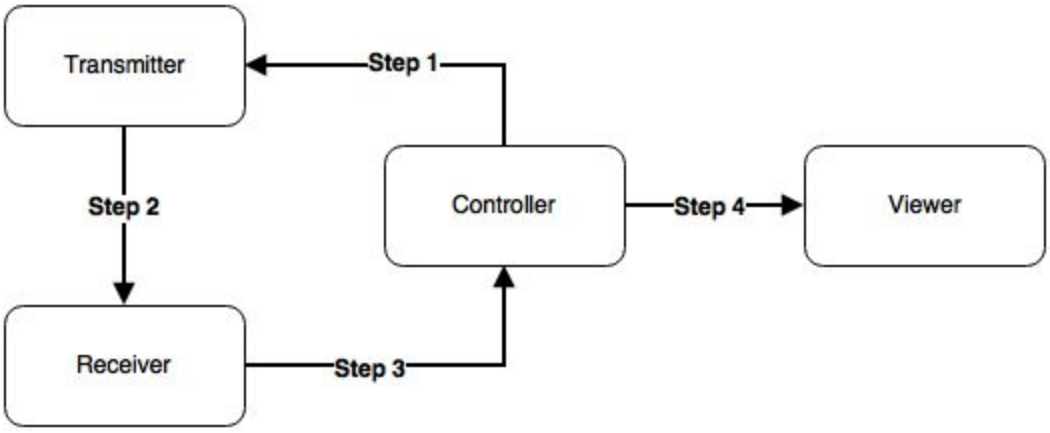

Figure 3: System diagram showing relationships between major components

2.2. **Transmitter**

The transmitter was designed using a 220 Ω resistor in series with a white LED. The LED is driven using a arduino PWM pin (PIN 3) alternating between high and medium duty cycles. The main idea with the transmitter was to conceal transmissions from the human eye, this meant that sending signals needed to be done quickly and without large changes to the LED's intensity. In order to achieve this, a default LED brightness was chosen (35 units), and a transmission LED brightness was chosen (255) in order to execute the software interrupt. An example of this can be seen in Figure 4.

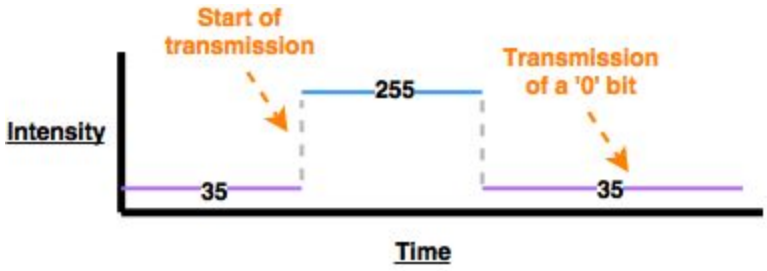

Figure 4: Graph showing ambient light intensity as well as interrupt case

Figure 4 also shows the use of interrupts to detect the start of a transmission. Every transmission begins with a digital HIGH value, and then relies on a timing sequence pre-defined to adhere to the limitations of the hardware. The timing sequence is a function of the receiver, specifically the total number of measurements multiplied by a delay, which results in a highly accurate system. The schematic for the transmitter can be seen in Figure 5.

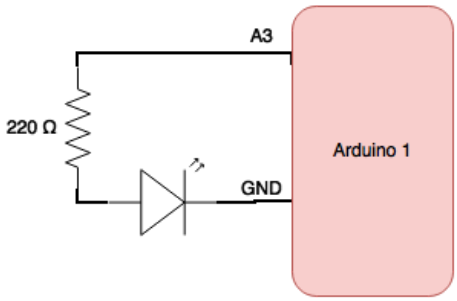

Figure 5: Schematic of the transmitter

The most important aspect of the transmitter is the ability to receive a message to transmit from the serial port. The transmitter is designed to block until a message is written to the serial buffer, at which time it is saved to a variable and transmitted. The values received from the buffer are bytes, and thus require formatting changes in order to use in PWM calculations. The transmitter PWM implementation can be seen below:

```
analogWrite(LED, ((int(msg) - 48) * 220) + 35);
```

The msg variable is storing the byte received from the serial buffer, which is converted to an int and subtracted from 48 (the subtraction is to account for the ASCII conversion). The resulting value (either a 1 or 0) is then multiplied by 220, which results in either a high or low output, and finally is added to the default value of 35.

2.3. Receiver

The receiver is arguably the most important component in the system, it performs 3 major operations. The first is polling the environment to determine an average intensity to base measurements on. This calibration is performed every 35s, but can be adjusted to better suit the operating environment. This can be seen in Figure 6.

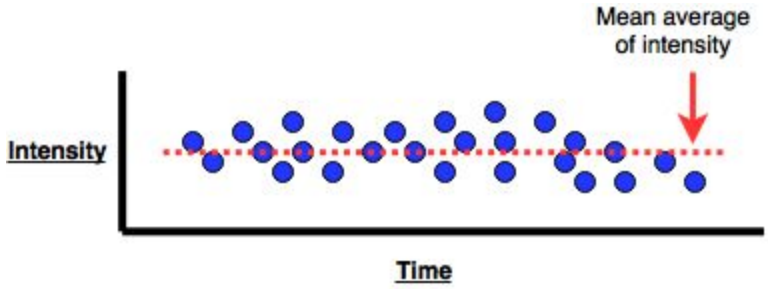

Figure 6: Graphical representation of mean average calculation for intensity

The second major operation is the recording and differentiation of readings into the ON and OFF categories to determine whether a 1 or a 0 bit was sent by the transmitter. The implementation method chosen favoured security and dependability over speed. Each bit is read 3 times, with each reading being 15ms apart. The 3 readings are then included in a mode average calculation to determine whether the transmitter sent a 1 or a 0. In order to find out what category each received value falls under, a predetermined intensity threshold is used. The threshold is an approximation of the intensity jump expected during the transmission of a 1. This can be seen in Figure 7.

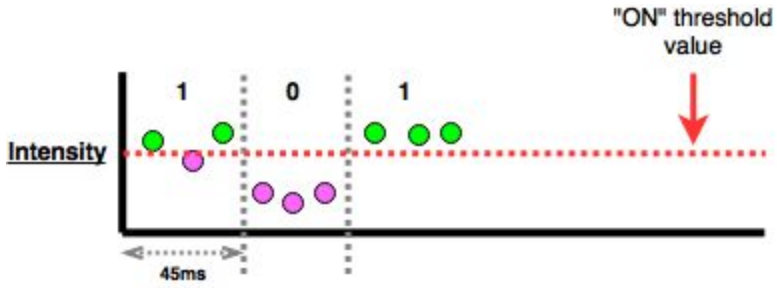

Figure 7: Graphical representation of mode average calculation for data

The final major operation performed by the receiver is the writing of the received data back to the serial buffer. This is done so that the controller can read the data, and convert the data back to ASCII, and send it to the appropriate node. The schematic diagram of the receiver can be seen in Figure 8. The receiver makes use of a photoresistor in a voltage divider with a 3.3 kΩ. The 3.3 kΩ provided the best readings, while higher resistors resulted in less sensitivity,

and lower resistor values subsequently resulting in higher sensitivity.

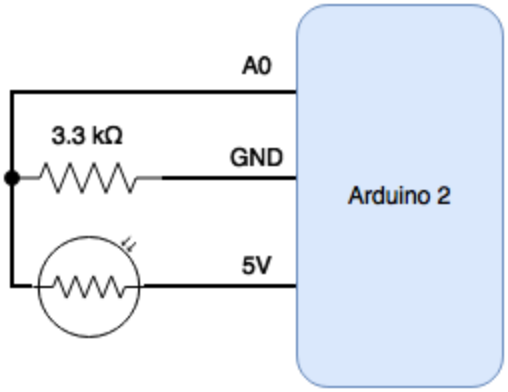

Figure 8: Schematic of the receiver component

2.4. Controller

The controller was developed to help create an easy and intuitive interface to the physical layer of Li-Fi. The controller was written in python, and makes use of the pyserial library in order to communicate with the arduinos. When initiated, the controller creates a connection with both arduino serial ports and prompts the user to enter a node ID and message. The node ID is a 3 bit unique identifier associated with a specific receiver. Despite only implementing one receiver for the demonstration, the infrastructure to support several nodes has been built in. The next step for the controller is to convert the entered message to a binary string. The function maps each ASCII character to a 7 bit long binary string. Once complete, the unique identifier associated with the node ID entered earlier is added to the beginning of the binary string. The current implementation supports 4 characters, resulting in a 34 bit string (counting the 3 bit unique ID).

After writing this string to the buffer of the transmitter, the controller waits for 2s before reading the buffer of the receiver. The binary string read from the receiver's buffer is split into two separate strings consisting of the message, and the unique identifier. Both are decoded back to the original ASCII values and saved to data.html. Each time the controller script is started, the data.html file is re-written. This functionality was implemented for the demo to emphasis the classes selected strings. Below are the functions used to convert the ASCII messages to and from binary strings.

```
# to binary
send_msg = bin(int(binascii.hexlify(send_msg), 16))
# to ASCII
nrec_msg = binascii.unhexlify('%x' % int((msg), 2))
```

2.5. **Viewer**

The final component of the system was the viewer. The viewer was designed as a method of posting messages that have been sent within the network. The implementation shown during the demo is a simple application running on node.js that prints the most recent messages sent and received. Future development on this component would see user interaction with the network implemented through the viewer so that messages could be written on the controller from the internet.

Having the system integrated with the internet provides great opportunities for feature expansion and integration of great tools and frameworks on the web. The viewer could be implemented as a android/ iPhone application that uses a solar panel module to passively detect signals being sent by Li-Fi enable environments. The view interface can be seen in Figure 9.

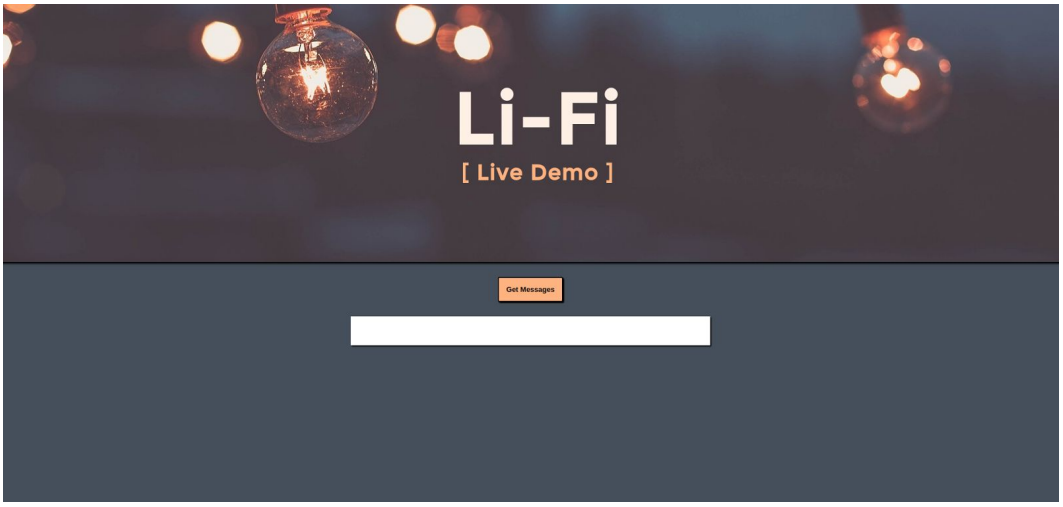

Figure 9: Web interface for Li-Fi network

3. Conclusion

The final implementation of the system successfully transmits and receives messages sent over light. This project was an opportunity to pursue a relatively new field and attempt to design a wireless network from scratch. Information taught in class was used when designing the physical and MAC layers of the network, especially when implementing error resistant implementation methods. The next steps for this project are to increase the transmission speed, and design a more interactive web interface. The completion of these would lend well to creating a larger scale model of the project, where more receiver nodes could be added to the network. This project was a great learning experience and a enjoyable way to end the semester.