Unidirectional Wireless Data Transfer Between Multiple Devices Using the Visible Light Spectrum

Arda ŞAHİN

Project Consultant: Kıvılcım Yüksel ALDOĞAN

INTRODUCTION

Nowadays, the functionality of the internet is almost limitless, and its technology is still changing and improving day by day. In the last decade, many new devices have been adapted to the internet, and this functionality has been accompanied by many new internet users. Many of these devices have to connect the internet wirelessly and our current technology supports it primarily by Wi-Fi (Wireless Fidelity) technology.

Although the Wi-Fi technology is easy to use with low cost, the rapidly increasing use of the Internet is filling up its frequency band quickly. If expressed by the numerical data, the growth rate of the internet is currently more than 11 users per second and the number of users in 2019 have been increased by 9% compared to 2018 alone, reaching 4.4 billion users **[1]**. With this problem, a new wireless technology with high bandwidth is being investigated, which may become an alternative to existing Wi-Fi technology.

One of these alternatives is the visible light spectrum, which has the bandwidth about 10000 times wider than the bandwidth of the radio waves, has no harm to human health and can be easily be managed by using LEDs and photodiodes. This idea was brought to the agenda for the first time in 2011 under the name of "Li-Fi" by Harald Haas from Edinburgh University **[2]**. In this technology, LED lamps, which can be switched on and off quickly, are used as a transmitter. After the data is digitally modulated, it is converted into an electrical signal and the LEDs transmit this signal as an optical signal. Since the brightness of LED lamps will change very quickly, they will not be distinguished by the human eye. On the receiver side, photodiodes, which accurately convert optical signals into electrical signals, are used. Finally, these signals are demodulated back into a binary system in the receiver circuit and thus information is transferred **[3]**.

OBJECTIVES

The main objective of the project is to have a unidirectional data transmission between a single transmitter and multiple receivers wirelessly using the visible light. The data to be transmitted include small sized files and real time audio samples. Although the subject of this project is inspired by the Li-Fi technology, the functionality of this project is more similar to Bluetooth technology since Wi-Fi technology requires complex bidirectional transmission protocols to access the internet.

In this project, both the transmitter and receiver sides are computers. There are two separate programs that has a User Interface (UI). With UI, the user can choose the files to transmit (maximum 4 files per transmission), configurate the transmission, create a voice recording and observe the transmission process. Both programs are created by the “Qt” software, which is a C++ IDE that contains the UI design tools.

The data transmission is done by the UART communication because of its simplicity and compatibility with the USB devices. The baud rate of the system is 460800, which is the highest baud rate value that the computer program and the USB-UART converter jointly support.

The transmitter side has a computer with USB-UART converter and an LED driver circuit. Right after the user starts the transmission process, the entire data will be read from the file and multiple encoding processes applied to the raw data. Next, the prepared data will be sent to a USB-UART converter, which is a bridge between the hardware and software parts. Then the UART signal is converted to optical signals by the transmitter circuit. The state of the LED reflects the exact opposite of the UART signal.

The receiving side has a receiver circuit that converts optical signals back to a UART signal. The main component for converting the optical signals back to the electrical signals is the photodiode. At the end of the receiver circuit, there is a USB-UART converter that reads the UART transmission. At the receiver application, decoding processes will be applied and finally, the raw data will be written on the newly created files. The block diagram of the transmission can be seen below:

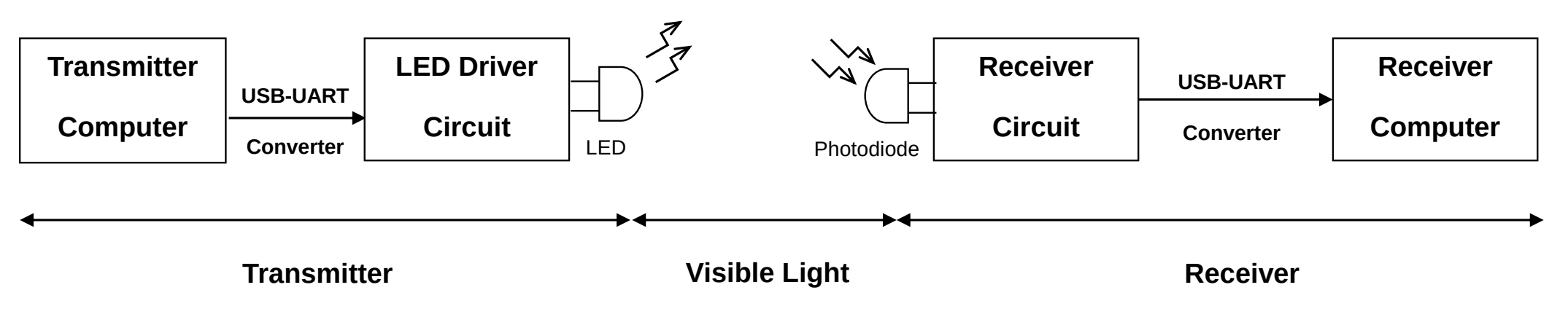
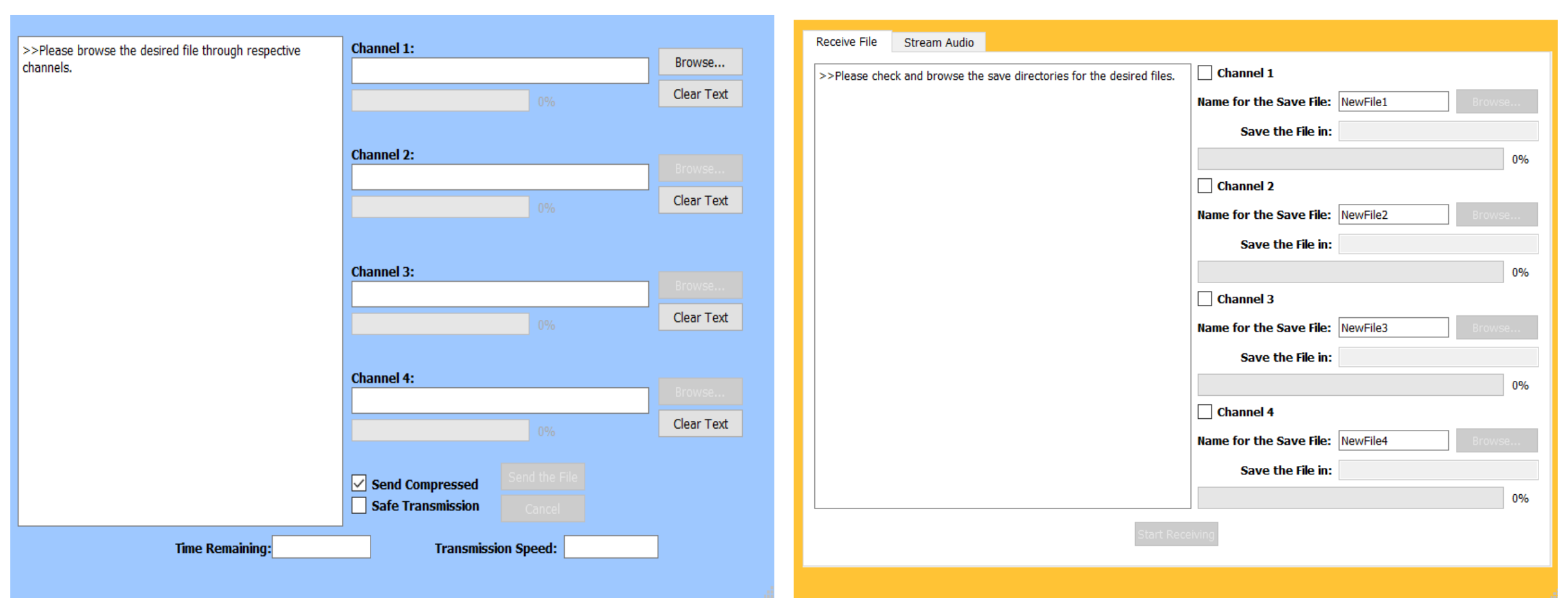


Figure 1: Block Diagram of the Project

SOFTWARE & HARDWARE DESIGN

The Software Part of the Project:

The software part of the Project consists of three main features. The first and main feature of the Project is the file transmission. Designed user interfaces for this feature can be seen below:


>>Please browse the desired file through respective channels.

>>Please check and browse the save directories for the desired files.

Figure 2: The UI Design for the Transmitter (on the Left) and Receiver (on the Right) of the File Transmission Feature

This feature supports up to 4 different files per transmission. With the user interfaces, the user can select some of the options for the transmission and can observe the transmission process with the event logs.

After reading the file, the transmitter does the following modifications to the raw data for making the transmission faster and more reliable:

Data Compression:

Compression is a source coding method to reduce the size of the target file. Once a file is compressed, it cannot be opened until it is decompressed. The area gained by compression depends on the contents of the target file and the algorithm used for the compression.

There are many algorithms available for the compression and for this project, the compression has been done by the "zlib" library. Zlib, is a specialized C / C ++ library for operations like file compression, saving compressed file, decoding compressed file etc. Zlib mainly uses Huffman and LZ77 encodings in its compression algorithms.**[4]**

(7,4) Hamming Coding:

Hamming Encoding is a channel coding method that is used for locating and correcting the possible error/errors in each symbol. For the Hamming code that has been used in this project, only one bit of error can be corrected for each byte of data. Hamming code uses redundant bits, which is called parity bits, to locate the error.

Hamming coding in this project starts by dividing 1 byte of data into two parts with 4 bits each. Two (7.4) Hamming codes are then prepared by using the separated 4-bit message bits. With the addition of the redundant bit, the amount of data to be sent at the end of this encoding will become double compared to the old one. A single byte of data to be sent can be visualized as follows:

p_1	p_2	p_3	d_1	d_2	d_3	d_4	n

Figure 3: A Hamming Code Block

p_n: Parity bits d_n: Message Bits **n:** Redundant bit

Data Encryption:

The data encryption that has been added in this Project increases the security of the transmission and reduces the amount of flickering in the light. Encryption will be applied to each byte according to the randomly created sequence of these four codings below. This random sequence is stored in an encryption file that both the transmitter and receiver application has.

Code 1: The sign of all the bits in a byte are inverted.

Code 2: Swaps the 4 bits on the right side and 4 bits on the left side in target byte.

Code 3: Only the 4 bits on the right side are inverted within the target byte.

Code 4: Only the 4 bits on the left side are inverted within the target byte.

The user interface of the next two features of the Project are shared in the same tabs below:

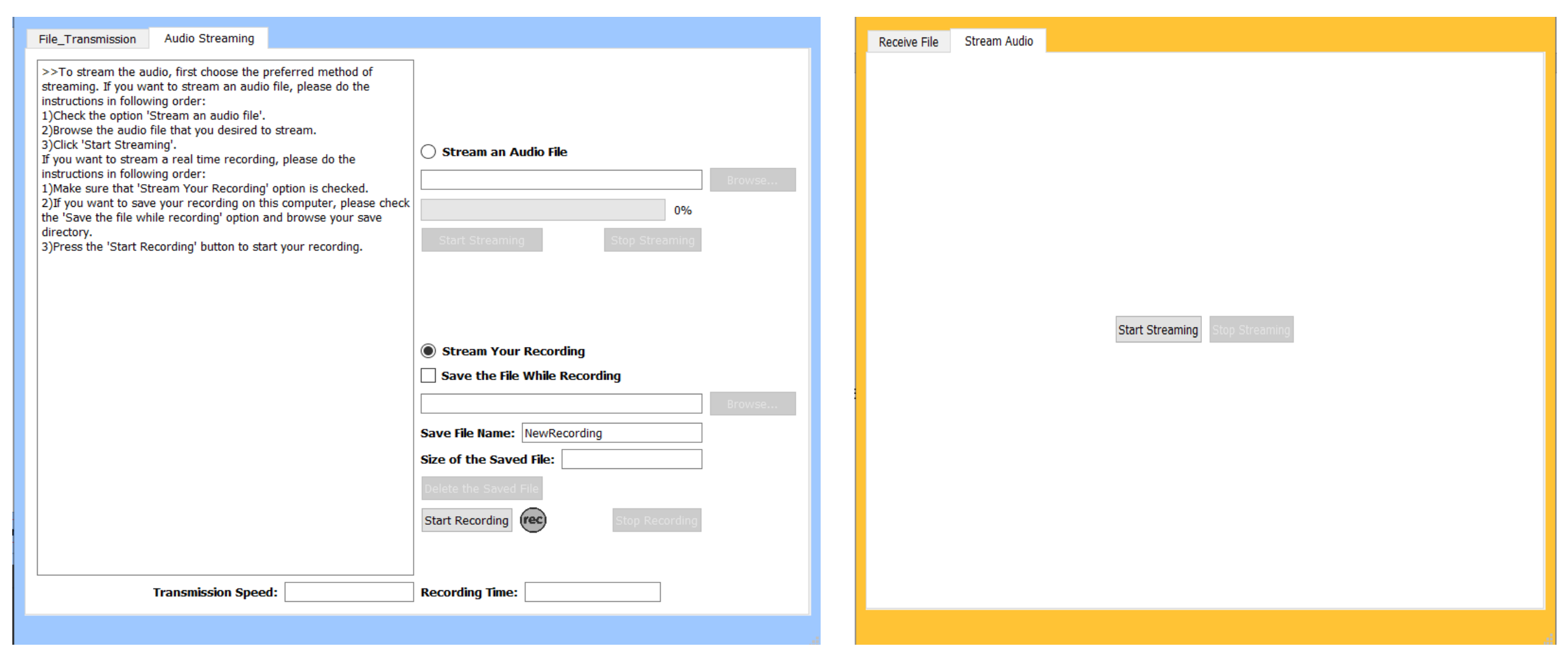

>>To stream the audio, first choose the preferred method of streaming. If you want to stream an audio file, please do the instructions in following order:
1)Check the option 'Stream an audio file'.
2)Browse the audio file that you desired to stream.
3)Click 'Start Streaming'.
If you want to stream a real time recording, please do the instructions in following order:
1)Make sure that 'Stream Your Recording' option is checked.
2)If you want to save your recording on this computer, please check the 'Save the file while recording' option and browse your save directory.
3)Press the 'Start Recording' button to start your recording.

Figure 4: The UI Design for the Transmitter (on the Left) and Receiver (on the Right) of the Real Time Audio Recording/File Streaming

These features allow user to play a wave file or their own voice recording in real time. In this configuration, computer at the receiver acts like a wireless speaker so it only accepts a specific format of an audio data. For this reason, the sampling rate of the wave files/audio recording must always be 11025, which is calculated by considering the transmission speed of the samples:

$$Sampling\ Rate = \frac{Baud\ Rate}{(8\ Message\ Bits + 1\ Stop\ Bit + 1\ Start\ Bit) * Bytes\ Per\ Symbol} = \frac{460800}{10 * 4} = 11520 \rightarrow 11025$$

The acceptable sampling rates for this Project can be seen below. When these sampling rates are encountered, the sampling rates will be fixed to 11025 with the data interpolation.

- **2756:** Upsampling by 4
- **3675:** Upsampling by 3
- **5513:** Upsampling by 2
- **7350:** Upsampling by 3 → Downsampling by 2
- **8269:** Upsampling by 4 → Downsampling by 3
- **11025:** Do not change
- **14700:** Upsampling by 3 → Downsampling by 4
- **16538:** Upsampling by 2 → Downsampling by 3
- **22050:** Downsampling by 2
- **33075:** Downsampling by 3
- **44100:** Downsampling by 4

The Hardware Part of the Project:

The hardware designs for this Project are shown below:

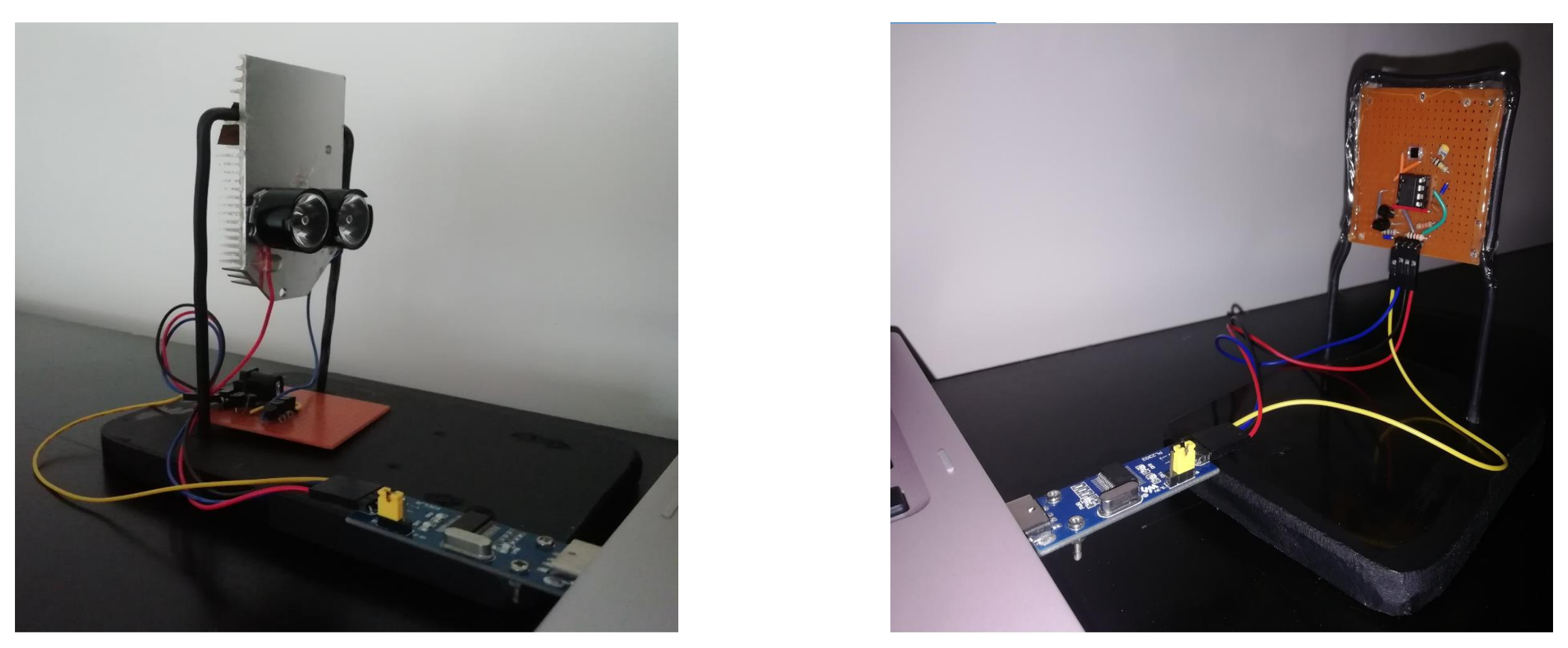


Figure 5: The Hardware Design of the Transmitter (on the Left) and Receiver (on the Right)

The transmitter circuit consists of two 3W rated power LEDs and a driver circuit that can handle 230.4 kHz square wave signals (with baud rate of 460800, maximum frequency that can be observed is 230.4 kHz.). The circuit diagram is shown below:

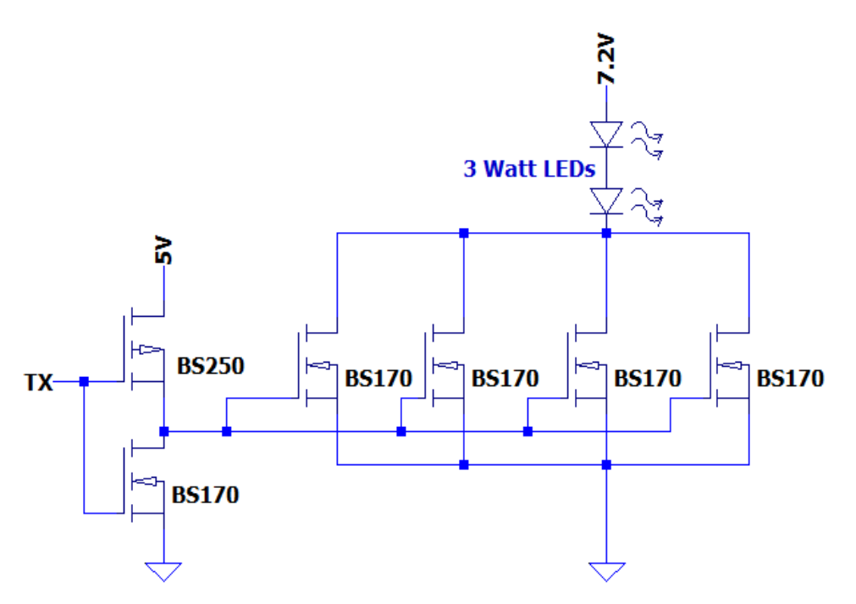


Figure 6: The Circuit Diagram of the Transmitter Circuit

The pulse shape from the TX pin consists of 2V as logic HIGH signal and 0V for the logic LOW signal. This voltage level is barely enough for turning the driver MOSFETs on but when they are opened this way, the internal resistance of these MOSFETs are too high for an efficient circuit. For this reason, a CMOS inverter is added to the circuit that makes the gate voltage switching between 0/5V which is much more efficient. The inversion will be inverted back to its original shape at the receiver circuit.

The circuit diagram for the receiver can be seen below:

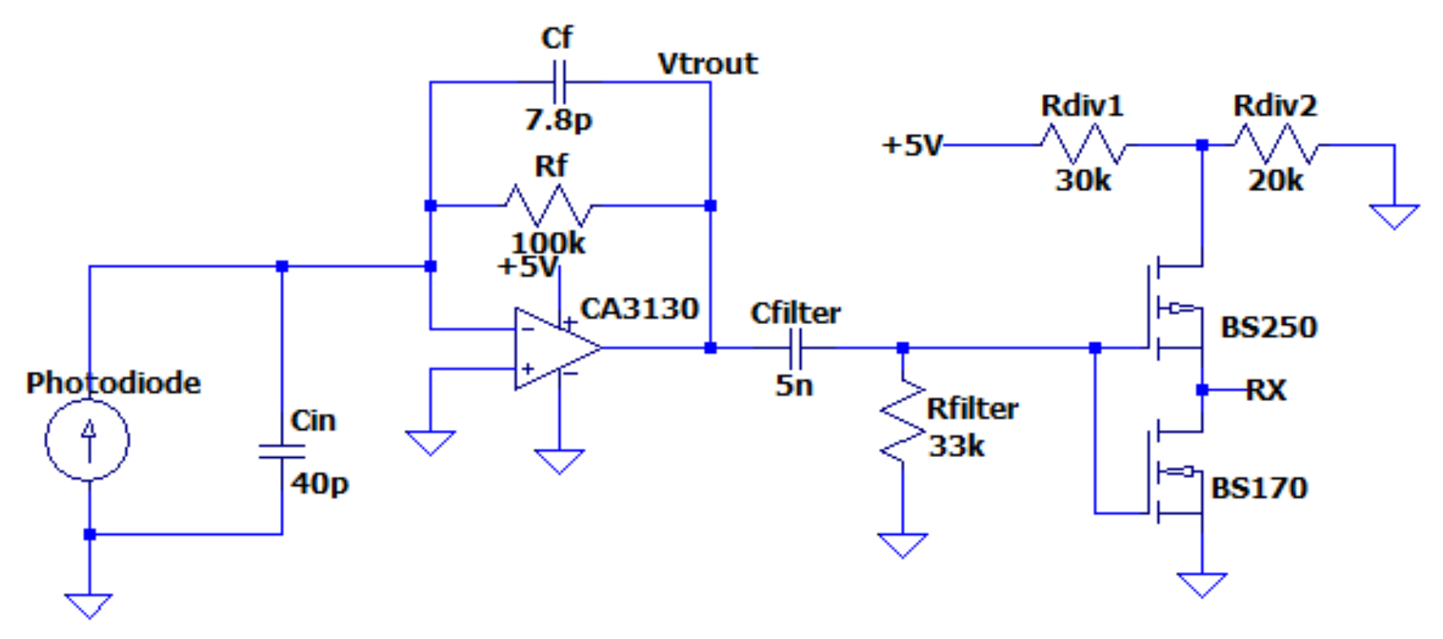


Figure 7: The Circuit Diagram of the Receiver Circuit

This circuit consists of a transimpedance amplifier, a high pass filter and a CMOS inverter. The transimpedance amplifier, converts the current signals from the photodiode into voltage signals. The high pass filter filters some of the low frequency noise components and the idle state of this Project. Finally, the CMOS inverter inverts the signal back to its original shape and shapes the waveform back into 0/2V signals.

RESULTS

Software Tests:

The software tests have been done by directly connecting two USB-UART converters together, which are connected into separate computers. This way any possible errors will be guaranteed to located on the software part. The configuration can be seen below:

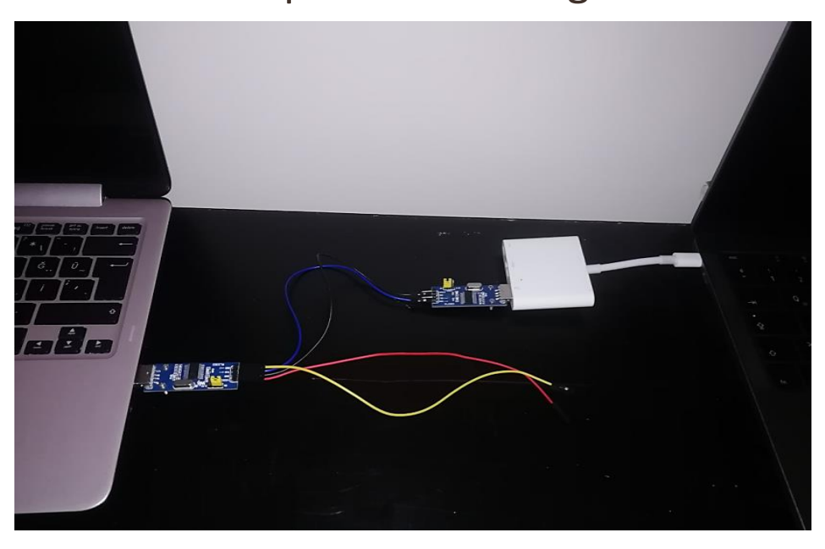


Figure 8: The Direct Connection Model

Hardware Tests:

The first result below shows the behavior of the gate voltage of the driver MOSFETs with the voltage of the TX pin:

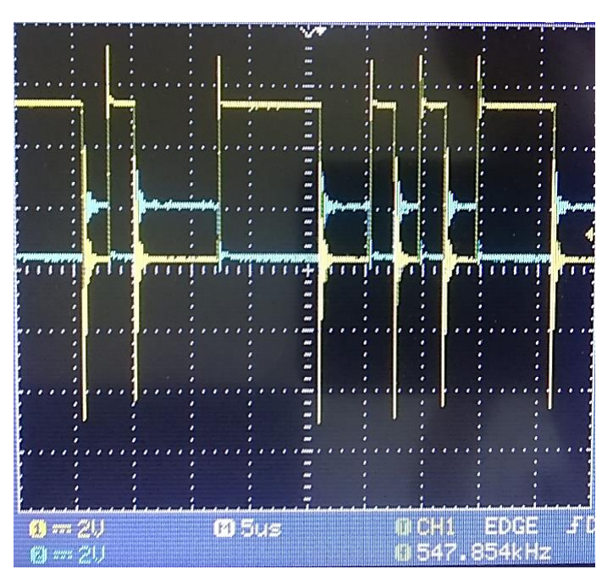


Figure 9: The Gate Voltage of the MOSFETs (Yellow) Versus the Voltage on the TX Pin (Blue)

It can be seen from the results that, the gate voltages on the MOSFETs follow the inverted version of the transmitted signal perfectly, which makes this section successful.

The next result below shows the RX pin at the receiver side with the TX pin of the transmitter side. Ideally the shape of these pulses should be the same.

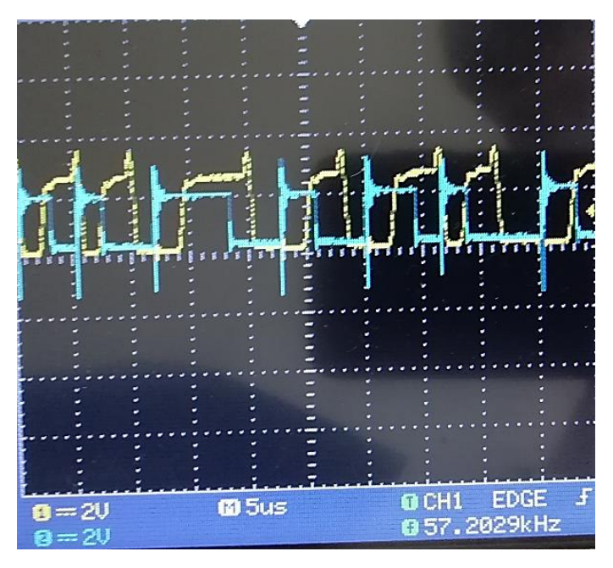


Figure 10: The Results From the TX Pin (Blue) Versus the RX Pin (Yellow)

It can be seen from these results that the voltage of the RX pin is almost the same as the delayed version of the voltage on the TX pin. The delay that has been created won’t be a problem since the transmission is asynchronous. The possible flaw of this circuit is the unbalanced logic HIGH/LOW times, which may cause a wrong decision.

CONCLUSIONS

Unfortunately, even though the software and hardware parts have been worked reasonably well on their own, the combination of these parts ware unsuccessful. The possible reasons are shown below. These problems may have been fixed if there is more time to fix these problems.

- When the software part opens the serial port, the light becomes closed in a short period of time right before the data transmission starts. This situation allows ambient noise to create false message signals. One way to solve this problem is to create a blacklist for the certain byte and mark the real blacklisted bytes by using the flag bytes. Other way to solve this problem is by ignoring a certain number of bytes at the receiver if the duration of this state is constant.
- Some bytes in the transmission are missing at the output of the receiver. The main suspect for this phenomenon is the transient response of the filter. If that is the case, the only solution for this problem is to add dummy bytes at the problematic regions, which will absorb the transient response and allow the actual data to be taken. The main problem for this solution is to optimize the number and location for stuffing these dummy bytes which will take a long time to handle.

REFERENCES

[1] The numerical data is obtained from: https://www.clickz.com/internet-growth-usage-stats-2019-time-online-devices-users/235102/

[2] The main information about the Li-Fi technology is obtained from the TED talk of Harald Haas: "Harald Haas: Wireless data from every light bulb". *ted.com*. Archived from the original on 8 June 2017.

[3] The information about how Li-Fi works is obtained from: https://lifi.co/how-lifi-works/

[4] The information about the compression algorithms in “zlib” are derived from: https://zlib.net/feldspar.html