Final Project Report : Data Transmission over Laser

Seth Hirsh
Lab Partner : Michael Hyland
Physics 111, Spring 2013
University of California, Berkeley

December 15, 2014

Introduction

Data transmission is a major component of almost all modern electronics. Most modern techniques rely on the transmission of digital data consisting of bits, which can be used to represent a variety of media, including text, sound images and video.

In this project, we present a method for transferring digital data over distances using a laser. In particular, we test this method by transmitting a color image from one computer over a distance of approximately 3 ft to another computer which then displays the received image.

The basic idea is that an output of a laser at any given time can represent a bit. Either the laser emits light, which corresponds to a 1, or the laser is off, corresponding to a 0. Thus, a series of pulses from the can represent a set of bits, and hence a complete set of digital information.

Design

The general design of this project involves three major components: software to convert the image into bits and then to voltages, circuitry which can emit a corresponding sequence of pulses, circuitry to receive the pulses, and software to convert the voltages back into an image.

Transmitting Software

An arbitrary JPEG image file can be converted using the *Read JPEG File.vi* into a array of $m \times n$ pixels where each pixel is represented by three values, one corresponding to the intensity of red in the pixel, one corresponding to the amount of green and one corresponding to the amount of blue. (On a subtle note *Read JPEG File.vi* actually converts the image into a 1d array where every third element is the red value of a pixel, the following pixel is the green value, which is followed by the blue value). This array can be then be passed to the *Transmit* sub vi which will be described

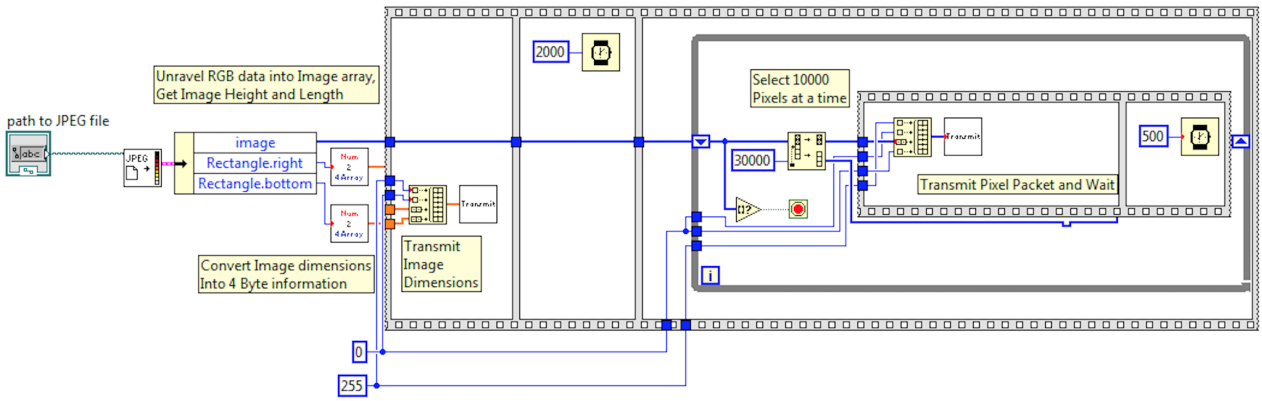

Figure 1: LabVIEW VI used to convert the image data in the form of bits into voltages.

shortly. Each of these three values range from 0 to 255 ($2^8 - 1$)and thus can be represented by 8 bits.

Before the signal is to be transmitted a similar signal is sent which contains information about the dimensions of the image. (In this case the width and height are accessed from *Rectangle.right* and *Rectangle.bottom* respectively). To utilize the same function for transmission (namely the sub vi *Transmit*), we must convert these dimensions into an array of values ranging from 0 to 255. The algorithm used to perform this conversion can be found in the *NumTo4Array* vi shown in Figure 2. First, four bytes (each byte being a value ranging from 0 to 255) are allocated for this value. If

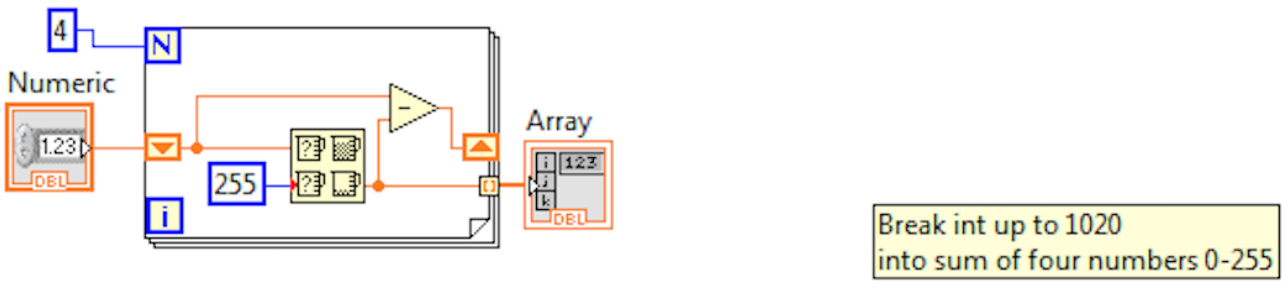

Figure 2: LabVIEW sub VI (*Num 2 4 Array*) used in Figure 1 used to convert the image dimensions into an array of 8 bit unsigned integers.

the height (or likewise the width) is less than or equal to 255 pixels in size then the first element array of set to the width and the rest of the bytes are set to 0. If the height is greater than 255 pixels but less than 510 pixels than the first element of the array is set to 255, the second element is set to #of pixels-255, and all remaining elements are set to 0. For larger dimensions the first two elements of the array are 255 and this process of subtraction is repeated. For example, 922 corresponds to the array $(255, 255, 255, 157)$. The advantage to this method is that converting this array back into the value of the height involves simply summing up the array values. On the other hand, this method limits the number of pixels for the height and width to $255 \times 4 = 1020$ pixels

each, although this limitation can be easily be removed by allocating more bytes.

In addition to this array data, a set of "triggering" bits (consisting of a byte representing 255 and one byte representing 0) are added to the beginning of the array. This corresponds to turning on the laser which can be detected and recognized by the detector as the time at which to begin recording data.

Next, the array of values ranging from 0 to 255 is passed to the sub vi (*Transit*), shown in Figure 3 to be outputted by the DAQ. In summary, this code iterates through the array converting each value into an 8-bit boolean array which are then concatenated and converted to 0's and 1's to form a 1d array of bits. These values are then converted to voltages using the scaling factor -4.5 V and then outputted by the DAQ as a n samples consisting of these voltages.

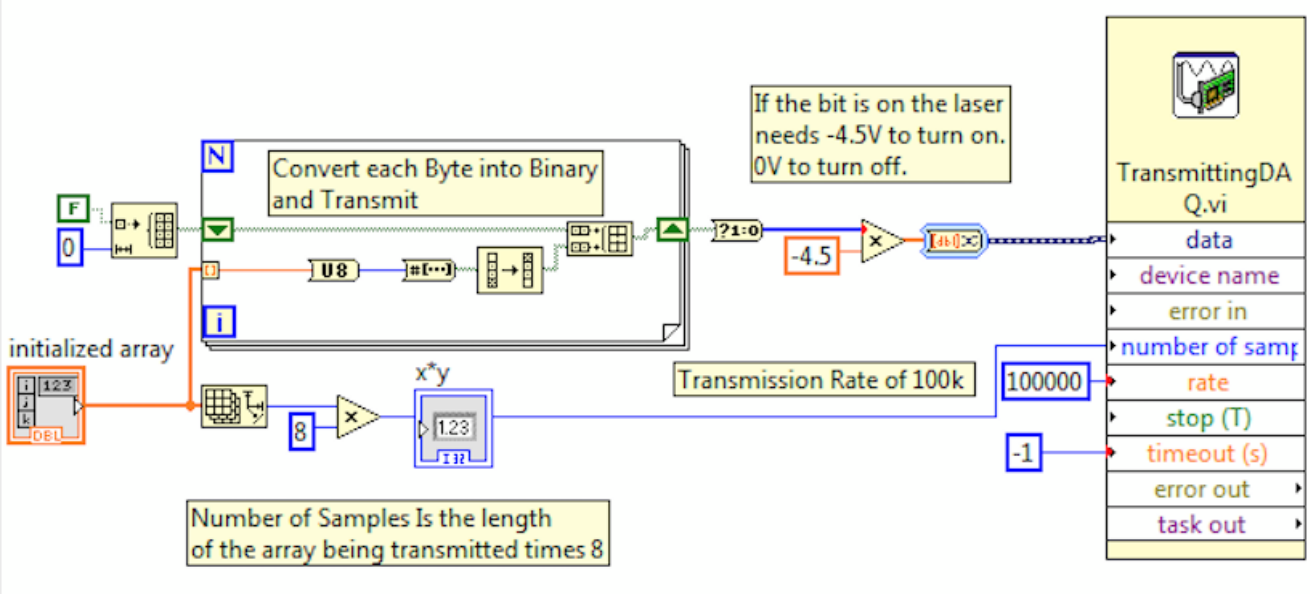

Figure 3: LabVIEW sub VI (*Transmit*) used in Figure 1 used to convert an array of integers into voltages.

Transmitting the image data is implemented in a very similar fashion to transmitting the height and width data. Much of the algorithm is based on the transmission and reception being synchronous. Thus, the data received was much less corrupted if, rather than sending the entire signal at once, only small packets of informations consisting of 10,000 pixels are transmitted at a time. In addition, to the leading 255 and 0 triggering bytes a following 0 byte is appended at the end to ensure that the laser is turned off in preparation for future triggering. Besides this, the wait methods are added to give time for the circuitry and detection software to prepare for the next packet of data.

Circuit Design

The circuits used for the project were relatively straightforward. To emit the signal a laser (taken out of a $3 laser pen) was directly connected to the DAQ. One limiting factor of the laser was its

finite rise and fall time. Figure 4 shows the output of a square wave from the laser at 1 kHz, 100 kHz, and 1 MHz. Besides some noise, the 1 kHz square wave looks identical to a square wave while the 1 MHz wave is signficantly distorted. Consequently we chose to use output at sample rate of 100 kHz for which the rise and fall times of the wave are small relative to the wave's period.

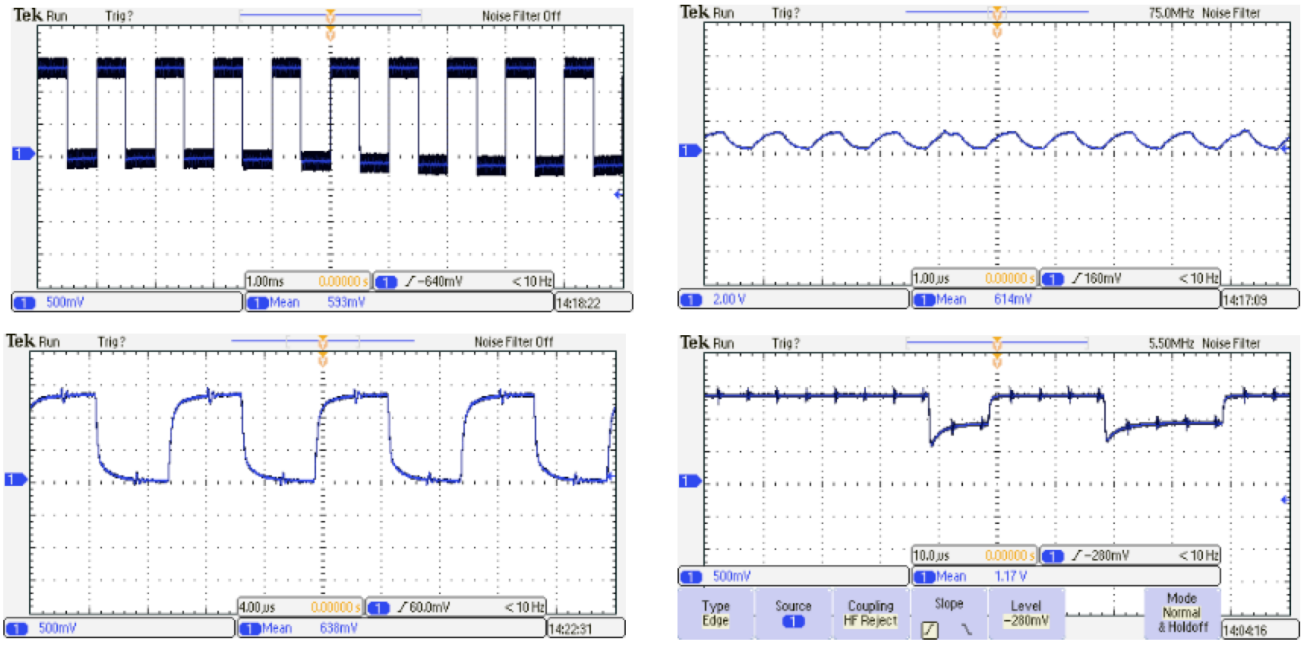

Figure 4: Top left: Waveform measured from output of laser using a 1 kHz square wave. Top right : Waveform measured from output of laser using a 1 MHz squar wave. Bottom left : Waveform measured from output of laser using a 100 kHz square wave. Bottom right : Output from laser using image data at an output frequency of 100 kHz.

The circuitry for the receiver is slightly more complicated and is shown in Figure ... On the left there is PIN photo diode which detects when the laser is emitting light. We used this type of diode due to its response rate of 20 ns (the photo diodes in the lab have a response rate of only 7 ms), which is more than fast enough to respond appropriately to a 100 kHz signal.

Using a reverse biased diode, as we would expect there is a 12 V (corresponding to a 0 bit) voltage drop across the diode. When the laser hit the diode the voltage dropped to 11 V (corresponding to a 1 bit). The voltage then passed through an inverting voltage follower. The only purpose of the follower was to measure the voltage at that point without worrying about affecting the current. The DAQ assistant in LabView is unable to trigger at a voltage as low as -12 V. Thus, we added a summing amplifier to increase the voltage range (which was from -11 V to -12 V), to values nearer to 0. By using $1k\Omega$ resistors as shown the resulting voltage is simply the negated sum of the two input voltages. Thus, the laser being on corresponded theoretically to an output voltage of -1 V ($-(-11V + 12V)$) and the laser being off corresponded to an output voltage of 0 V ($-(-12V + 12V)$). Measuring th voltages, we found that the laser being on actually corresponded to -1.6 V and the laser being off corresponded to -0.6 V.

This voltage was then passed to the receiving DAQ to be manipulated in LabView.

Receiving Data

The main piece of detection software code can be found in Figure 5, which has similar functionality to the transmitting code but executes in reverse. In particular, the sub vi used (*Detect*) is shown in

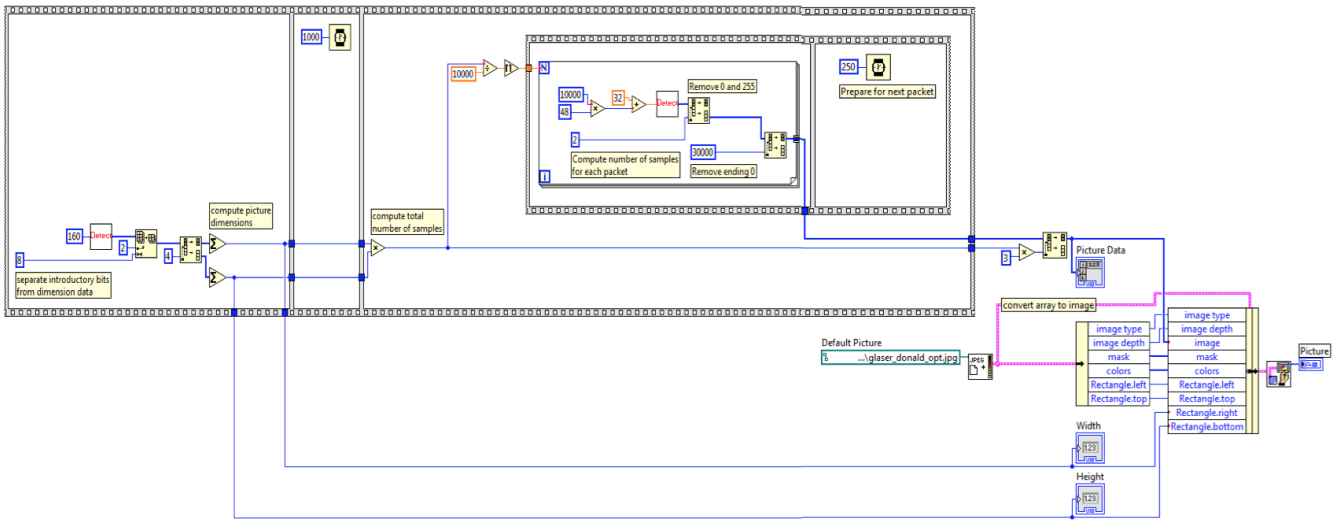

Figure 5: LabVIEW VI responsible for converting the received signal into the corresponding image.

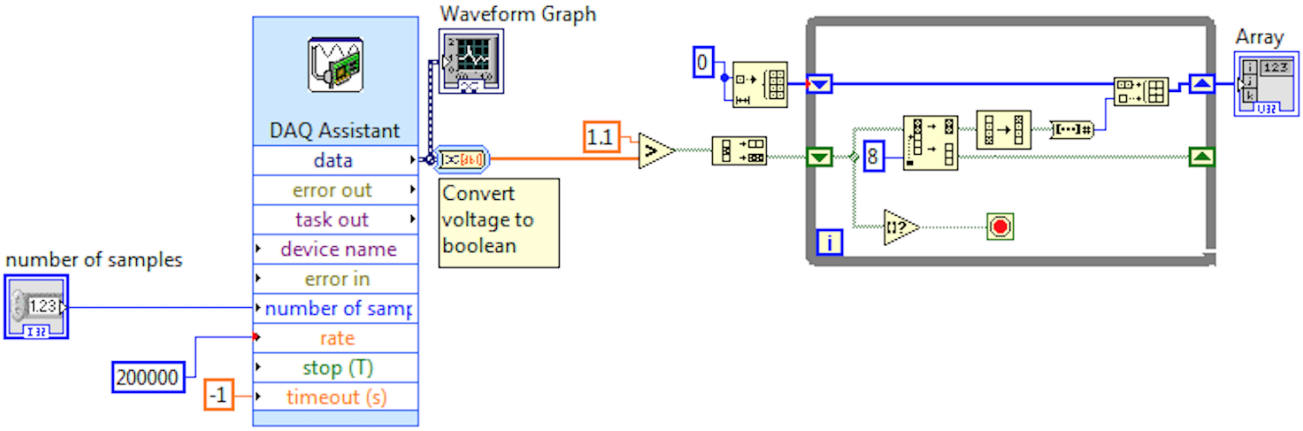

Figure 6: Sub VI (denoted as *Detect* in the "mother" VI) which convertes the received signal into an array of integers ranging from 0 to 255.

Figure 6. The DAQ is set to trigger when a rising voltage is detected. This value is then compared to fixed threshold voltage (in our case this value is 1.1 V). If the voltage measured is greater than this value, the corresponding pulse is measured to be a 1. Otherwise, it is recorded as a 0. Next

every set of 8 bits is converted to an integer (ranging from 0 to 255) and concantenated into an array returned to the "mother" vi. Note that the detecting DAQ samples at a rate of 200 kHz, twice the rate at which the data is transmitted. This is done so that the voltage is not measured in a rising or falling of pulse, but rather on the flat region of the pulse.

Returning to the "mother" vi, it is fairly clear that in the first element of the flat sequence structure, the 160 (2 samples per pulse $\times$ 8 bits per integer $\times$ 10 integers) samples are measured, the first integers (the 255 and 0) are thrown out and the last 8 integers are split in two and summed to compute the width and height.

Likewise, in the third element of the flat sequence structure, the DAQ measures the pulse voltages iterating through the necessary number of packets and concantenating the data. In addition, the leading 255 and 0 and the trailing 0 are removed from the data. With the data in the format of a 1d array of 8 bit integers the data can be easily converted to an image by bundling the image data and the *Draw Flattened Pixmap.vi*. Note that some elements used in the bundle are taken from a sample JPEG file. These values mainly consist of zeros and empty arrays and can be found in any JPEG file.

Results

To test the setup, this program was run using a variety of images. Every transmitted image appeared identical to the original image. To confirm this we converted the original and transmitted images into the corresponding arrays. As expected these arrays were identical. An example output can be found in Figure 7. The only case in which the transmitted image appeared corrupted was when an object was momentarily placed in between the laser and the receiver. In this case a black line would replace the blocked information, but once the object was removed all later information was recovered.

Setbacks and Solutions

The previous sections have described the final design of the project. However, throughout development there were many setbacks. Due to the rapid rates ($\approx$ 100 kilobits per second) many of these setbacks result from timing issues and are described below.

1 Sample vs. N Samples

In the original design, we wanted the receiving DAQ to measure one sample at a time in a while loop. This would allow us more control over the received data, for example allowing us to construct more sophisticated triggering techniques. An initial setback was that 1 sample on demand does not occur at a fixed rate. As can be seen in Figure 8 a fixed a sample rate can result in a wide variety delays between samples. To combat this issue hardware timing can be used, which results in much greater precision with timing. Even when hardware timing is used, LabView is unable to execute 1 sample at any rate above 1 kHz. This is a result of the fact that each time a sample is measured

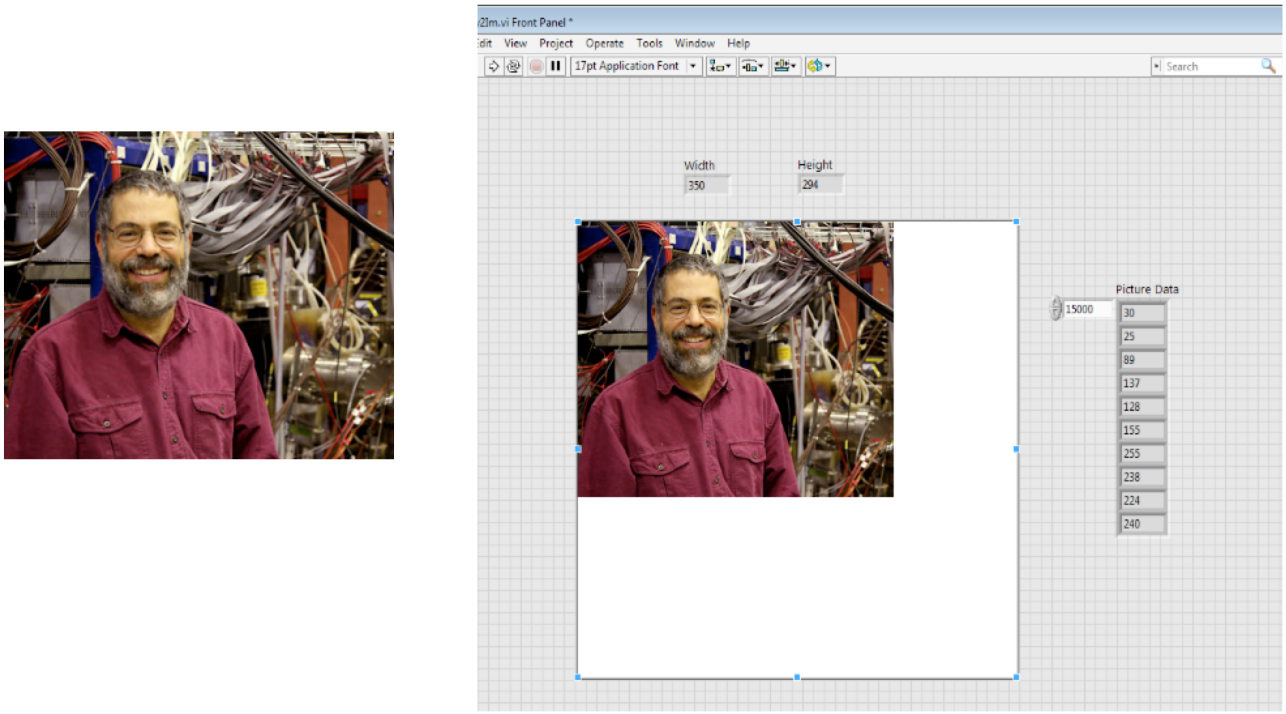

Figure 7: On the right is the transmitted version of the image on the left. Note that they appear identical.

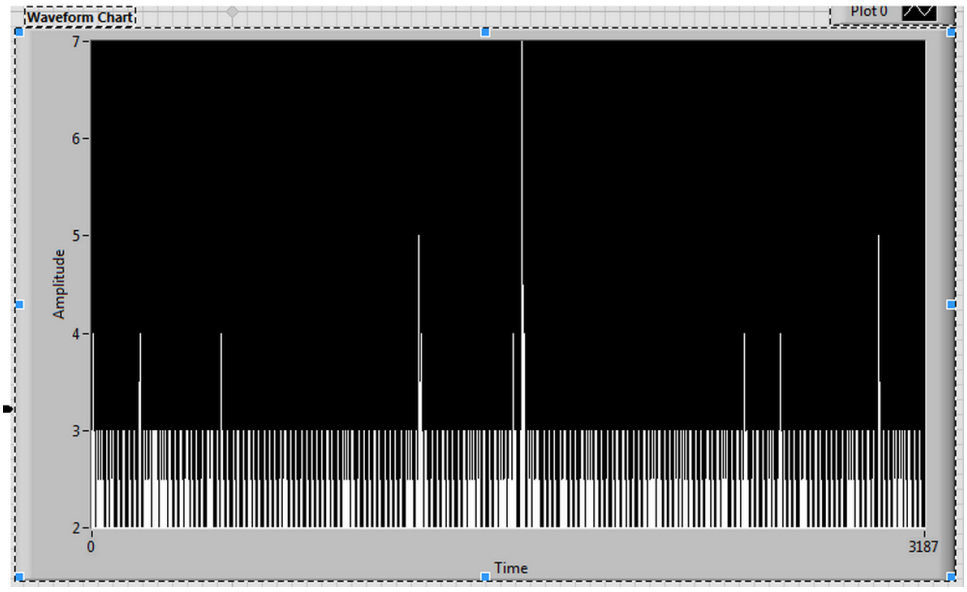

Figure 8: Distribution of wait times for a 2 ms time delay. Note the spread in values, which is intolerable for the precision necessary for this project.

the DAQ must be reinitialized, since the scope of the DAQ is only the while loop which contains it.

One solution to this problem is to replace the DAQ Assistant vi with its constitutent parts and place the instantiation of the DAQ outside of the loop. Although this method works in theory, the internals of a DAQ are quite complicated and it is preferable to only work with the DAQ assistant.

Instead, we resorted to using the "n samples" option predetermined by the packet length and simplfying the triggering mechanism (described below). Thus, the detector no longer ran continuously and would only measure a fixed number of samples.

Triggering

Knowing when the signal should begin and end were some of the most critical aspects contributing to the success of the project. Since all packets of information were fixed in length (the size data was 80 bits and each packet contained 10,000 pixels of information) and transmitted at a constant predetermined rate there was no ambiguity in when the signal should end.

However, determining when the signal should begin was a bit trickier. The initial plan was to have a four byte code, containing four 8-bit integers, that was inputted into the receiving software beforehand. The receiving software would then measure one sample at a time, and compare the last 32 samples measured (corresponding to 4 bytes) to the four integer "code". If these two sets of samples matched, the receiving software would "trigger" and immediately begin recording the image data.

This method for triggering is perfectly reasonable as long as the data received is not corrupted. As discussed in the **1 Sample vs. N Samples section**, this method depends on the ability to process 1 sample at a time, which was not an option with the current implementation of the code.

To remedy this, we used the option on the DAQ for a rising trigger depending on the input voltage. In other words, if the voltage measure rose above 1.1 V, the program began recording the data. As can be seen from the results this method of triggering was very successful. However, it depended on the fact that the laser was turned off before the triggering. Otherwise, false triggering would occur, resulting in incorrect determination of the image size. To ensure that this never occurred an extra 0 was appended to each packet of data telling the laser to turn off.

Different Clocks

Now consider if the emitting frequency of the data emitted by the laser was 100 kHz while the data was received and measured at a rate of 100.001 kHz. After 100,000 bits had been transmitted the receiver and emitter would be off by 1 bit. A 500×500 pixel image contains approximately 6 million bits of information. Thus, for no data corruption to occur the sample rates of the emitting DAQ and receiving DAQ must agree to incredible amount of precision. As expected, after approximately 240,000 bits (10,000 pixels) of data had been transmitted the difference in the clocks used in the DAQs resulted in the data becoming corrupted.

To remedy this, we packeted information into sets of 240,000 bits, with each packet containing the trigger bit and ending 0 bit. In addition, a wait time of 0.5 s was added to the transmission to allow time for the detecting software to prepare for the next packet. Although this method was succesful, the wait functions did noticeably increase the amount of time necessary to transfer an image.

Conclusion and Suggestions for Future Projects

In this project, we were able to successfully transmit an image over a laser by converting the image into bits. In addition to learning good data transmission techniques, we learned much about the workings of LabView as we had to execute the programs with very precise timing. Due to the lack of corruption in the transmitted image our resuts provide a stable method for data transmission which can execute at reasonable speeds.

For further improvement we suggest using a more lasers with shorter rise and fall times for higher (on the order of MHz) frequencies. In addition, we suggest trying to transmit a variety of digital media over the laser, including music, text, and video. Last, we suggest determining methods for reducing the times used by the "wait" methods between the transmission of packets, since for large files this time delay would become unwieldy.

References

[1] Fajans, Joel. "Physics 111 Instrumentation Laboratory." *Physics 111 Instrumentation Laboratory*. 2014. http://instrumentationlab.berkeley.edu/.

[2] "LabVIEW System Design Software." *NI LabVIEW*. Web. 15 Dec. 2014. http://www.ni.com/labview/.

[3] "Principles of Data Communications: Basic Concepts and Terminology." *Synchronous Transmission*. Web. 15 Dec. 2014. http://www.sqa.org.uk/e-learning/.