Class:	EE 320L		Semester:	Fall 2018
Points		Document author:	**Bryan Kerstetter** **David Santiago**	
		Author's email:	**kerstett@unlv.nevada.edu** **santid4@unlv.nevada.edu**	
		Document topic:	**Final Project**	
Instructor's comments:				

Digital Phototransistor Optoisolator and Its Respective Applications

1. Introduction / Theory of Operation

This project regards the design and implementation of a Digital Phototransistor Optoisolator (DPO); as well as its respective applications. An optoisolator is an optoelectronic device that transmits signals between two separate electrical circuits by light. Optoisolators consist of a means to convert an electrical signal to a light signal and the light signal back to an electrical signal. On one end, the input signal of the DPO drives a LED. While on the other end, a phototransistor detects the light emitted by the LED. Circuitry then converts the light signal back to an electrical signal that serves as the output signal of the DPO. Where both the input and output signals of the DPO are digital signals. The DPO also serves as a level shifter. The DPO's input can be within a range of voltage levels; while, its output signal level can be set within a specified voltage range. Three derivative projects have been developed to demonstrate a few applications of the DPO. The first two projects concern the construction of a Bidirectional Optical Link and a Smart Switch, respectively. The third project proves that transmission of data through the DPO can occur with complete electrical isolation of the sending and receiving circuits.

2. Description of Experiments

2.2. The Digital Phototransistor Optoisolator

Optoisolators require a light source and a light sensor. Optoisolators are generally classified by its source-sensor pair. The DPO can be classified as an LED-phototransistor optoisolator.

The design of the DPO required the design of two separate circuits. The first circuit's purpose is to convert an electrical signal into a light signal. The second circuit's purpose is convert the light signal back into an electrical signal. These two circuits were implemented on two separate PCBs that together, form the DPO. The two PCBs were the LED Driver and the Phototransistor Board (PTB). The LED Driver would then be situated in a parallel manner such that there is an optical channel between the two boards. The general concept regarding the construction of the DPO can be seen in figure 1.

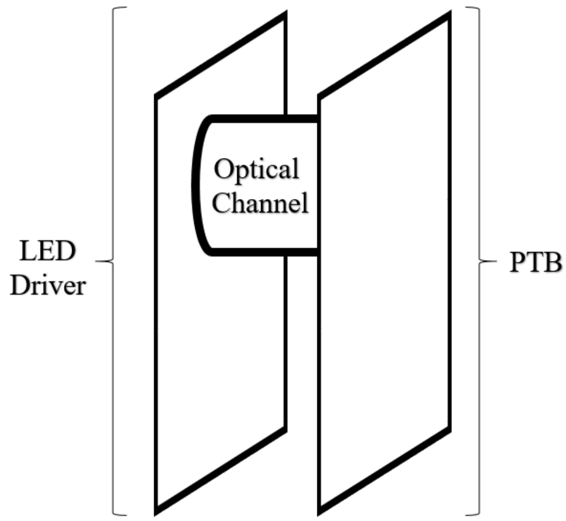

Figure 1

2.2.A. LED Driver

<u>Lighting an LED</u>

Generally, when using a LED, a current limiting resistor is used to determine the current flowing through the LED. One drawback of this approach is that as the supply voltage varies, the current following through the LED, varies as well. Another approach is having a current source supply current to the LED. A current source allows the current through the LED to remain constant. Figures 2 and 3 show a simulation regarding the two approaches described to light an LED.

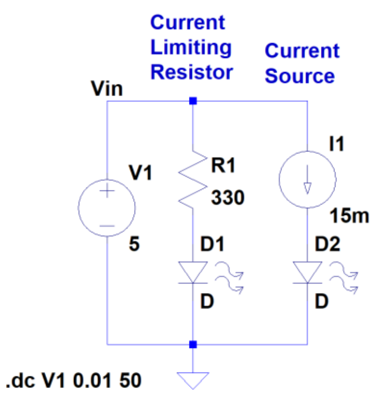

Figure 2

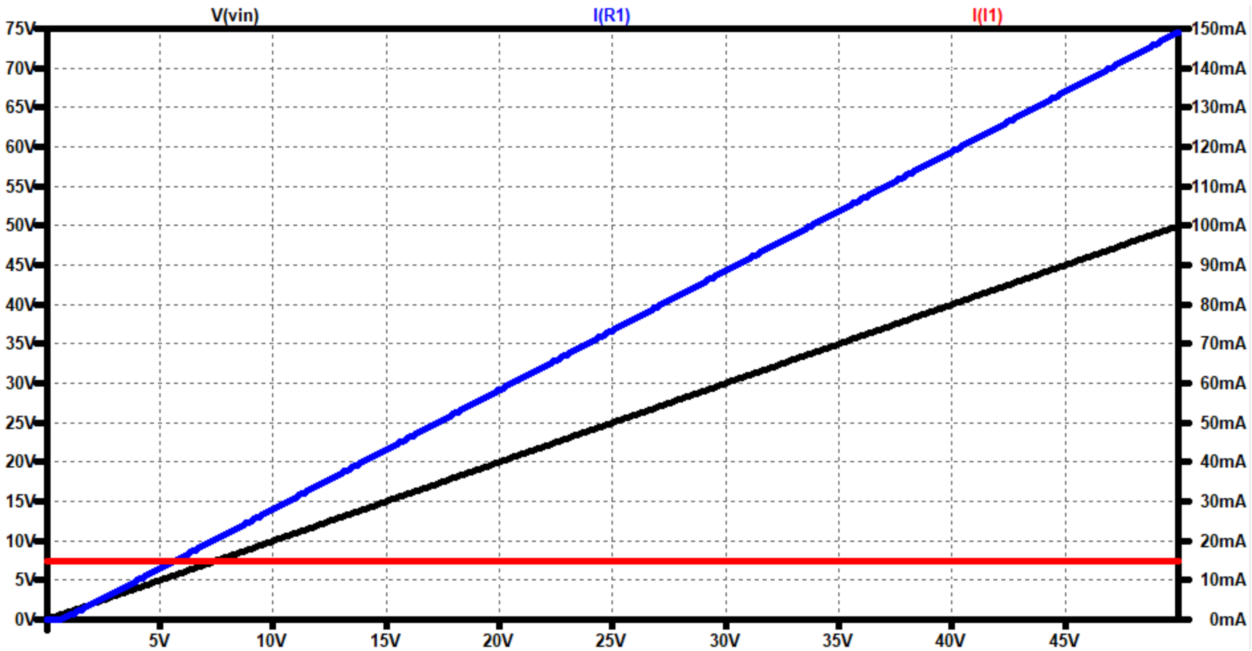

Figure 3

<u>LED Driver Circuit</u>

To light the LED on the LED Driver we will use the second approach. A LM334 IC can be used to create a constant current source. This will allow us to achieve similar results as seen in the hypothetical situation in figures 2 and 3. The LM334 can also be used to create a temperature sensor. Therefore, temperature becomes a variable to consider when designing an LM334 current source. Certain implementations of a LM334 current source, results in a current source that varies with current. However, a current source that varied with temperature was not desired. Pages 8 and 9 of the LM334 data sheet detail how to make a zero temperature coefficient current source. A zero temperature coefficient current source allows a current source to be not dependent upon temperature by nulling the effect of temperature change.

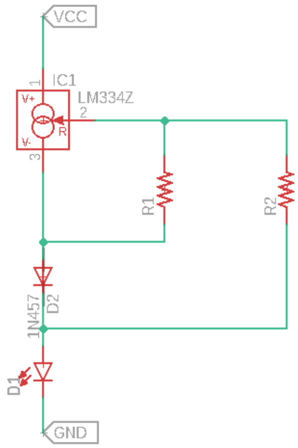

Figure 4

Figure 4 details the circuit implemented on the LED Driver. The selection of the diode D2 is very important due to the diodes temperature coefficient. D2 can be a 1N457, 1N914, or1N4148.

A potentiometer could have been placed where R1 or R2; giving you variable current source. However, this route was not chosen because if the LED was enclosed the potentiometer could be accidently adjusted such that the LED burns out without any obvious indication. Therefore, fixed resistance values were chosen instead.

A PCB was then designed to accommodate the LED Driver. A copper ring was created such that a copper tube could be soldered on; creating an enclosed optical channel.

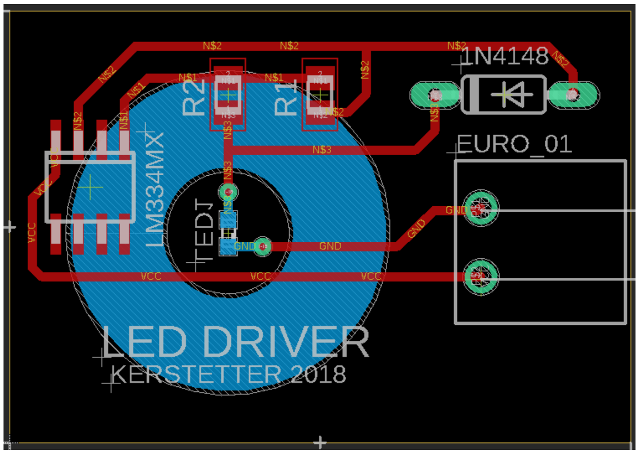

Figure 5

Figure 6 Fabricated LED Driver from Gold Phoenix (top side)

Figure 7 Fabricated LED Driver from Gold Phoenix (bottom side)

<u>LED Driver Example</u>

The procedure of calculating R1 and R2 is detailed in the LM334 datasheet (page 8-9). In summary:

$$I_{set} \approx 5\ mA = \frac{0.134\ V}{R_1}$$
$$(R_1)5\ mA = 0.134\ V$$
$$R_1 = \frac{0.134\ V}{5\ mA} \approx 26.8\ \Omega$$
$$R_2 = 10(R_1) \approx 268\ \Omega$$

To supply the LED with ~5mA, our calculations have determined resistance values to be:
R1a = 26.8 Ω
R2a = 268 Ω

Due to what was readably accessible, the following resistance values were then chosen:
R1b = 27 Ω
R2b = 267 Ω

We created an LTspice model of the LED Driver and then set R1b and R2b.

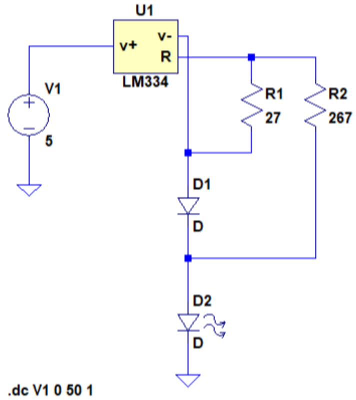

Figure 8

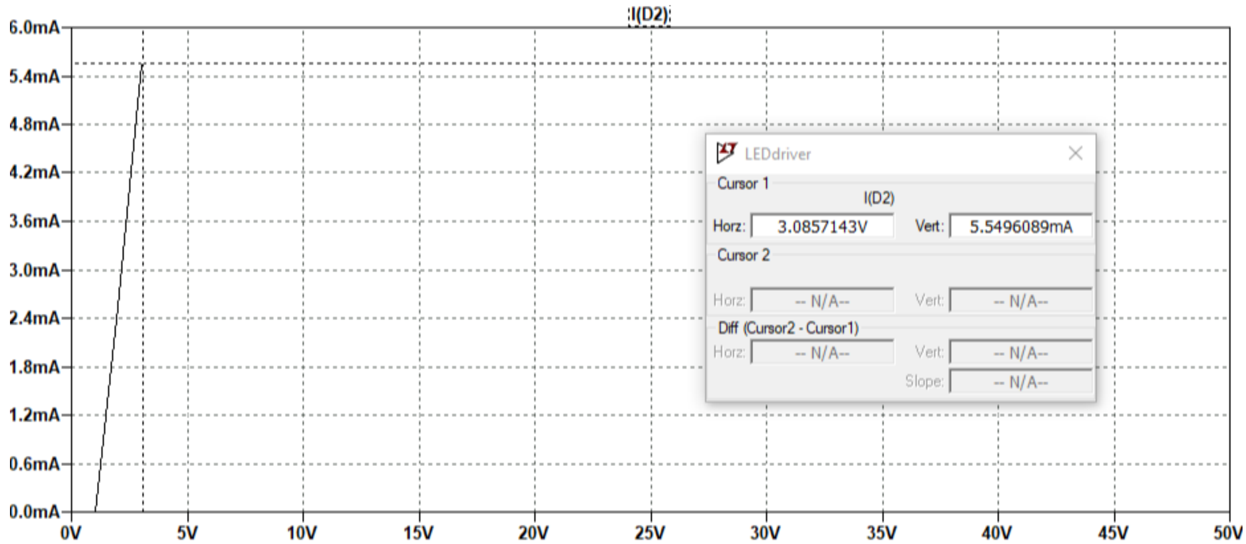

Figure 9

Figure 9 plots the current/voltage relationship of our LED Driver. In the simulation, 3.09V and above supplies the LED with 5.55 mA. The LED driver was then soldered together so that its performance could be experimentally examined.

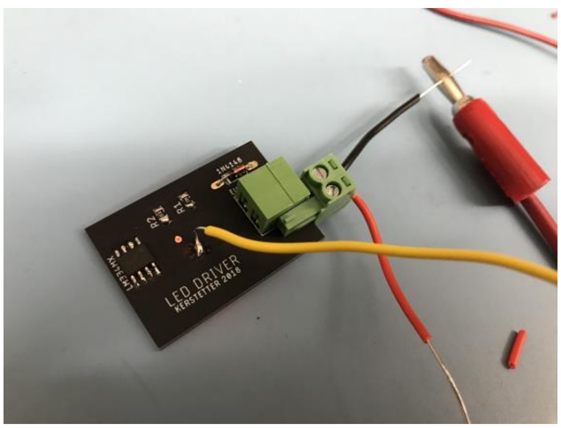

Figure 10

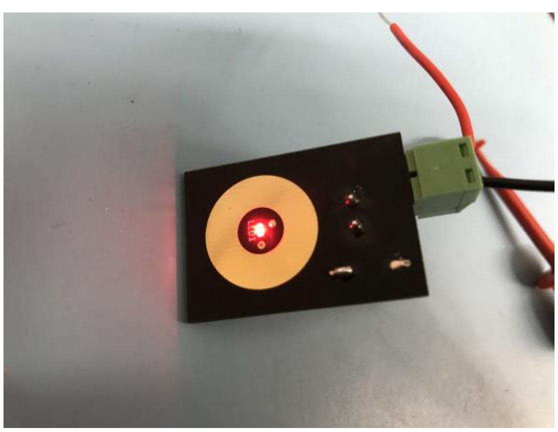

Figure 11

As the supply voltage increased the current through the LED rapidly increased until 4.5V. After the 4.5V volt threshold was reached the current through the LED increased at a slower rate as the supply voltage increased. The highest voltage tested was 31.6 V. Which was the max single channel voltage on the lab power supply.

V	I
0 V	0mA
4.5 V	5.394 mA
31.6 V	5.507 mA

Figure 12

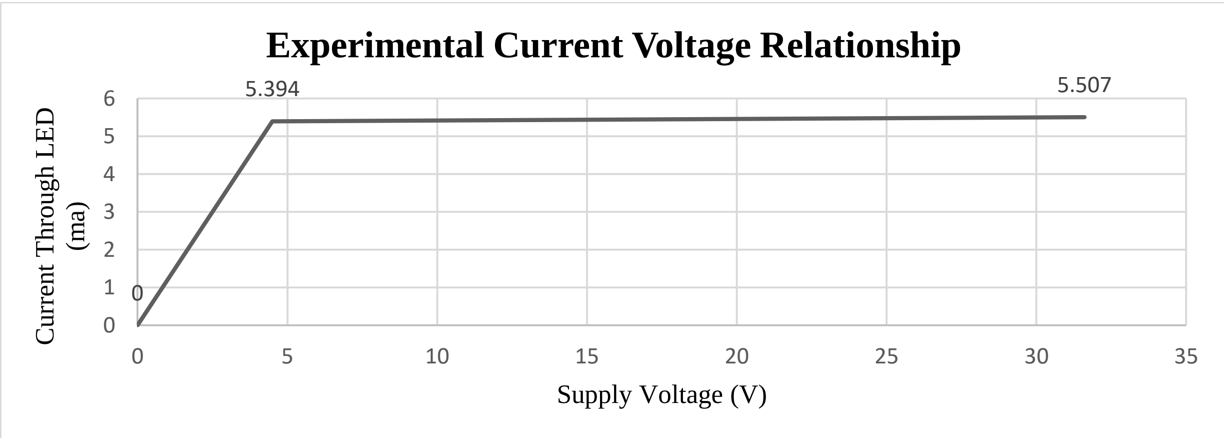

Figure 13

According to the LM334 data sheet the maximum input voltage is 40V. It should be noted that if the set current is too high; higher voltages have ceased the LED Driver non-functional. The LED Driver is proven to provide near non-changing current to the LED. Thus, allowing the LED to be lit at a consistent brightness for wide input voltage ranges. This key factor is essential to the function of the DPO as a level shifter.

2.1.B. PTB

The PTB contains the light sensor circuitry. Figure 14 shows the PTB circuit.

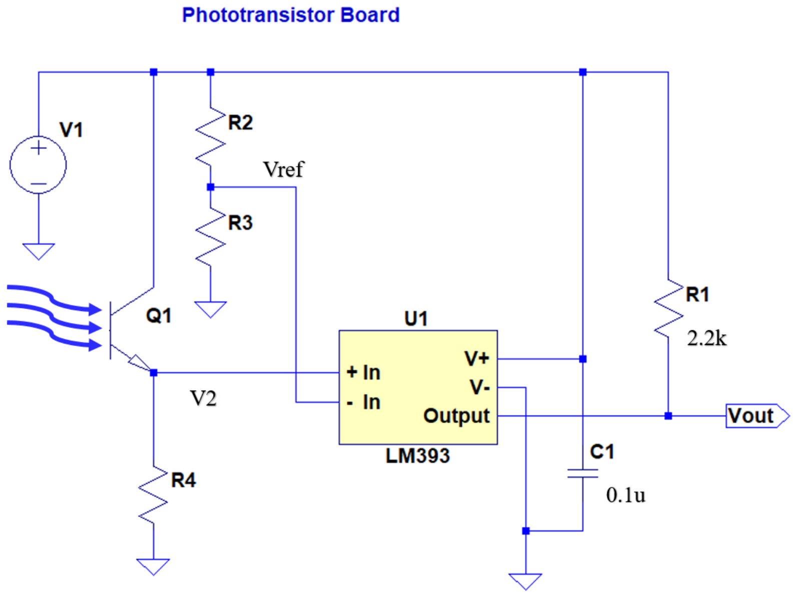

Figure 14

The PTB contains what makes our optoisolator digital. The design goal of the PTB was to output active high when the PTB detects a certain light intensity and active low otherwise. The central piece of the PTB circuit is the LM393 comparator. The comparator is what converts an analog light intensity sensor to a digital light intensity sensor. How the PTB behaves is based upon the resistance values of R2, R3, and R4. R2 and R3 simply form a voltage divider that creates a reference voltage Vref. R4 determines the sensitivity of the PTB to light. A R4 larger resistance value equates to greater PTB sensitivity. When V2 is greater than Vref the PTB outputs active high. To the contrary, when V2 is less than Vref the PTB outputs are active low. The light signal from the LED after the falling edge of the input of the DPO is a decaying voltage level. The comparator ensures that there is a clean falling edge in the output signal.

Figure 15 shows the PCB that was designed for the PTB. A copper ring was created such that a copper tube could be soldered on; creating an enclosed optical channel. We initially ordered our PCBs from JLC PCB. Unfortunately, we received PCB that were fabricated without the solder mask layer. We later ordered from Osh Park and we received PCBs of great quality.

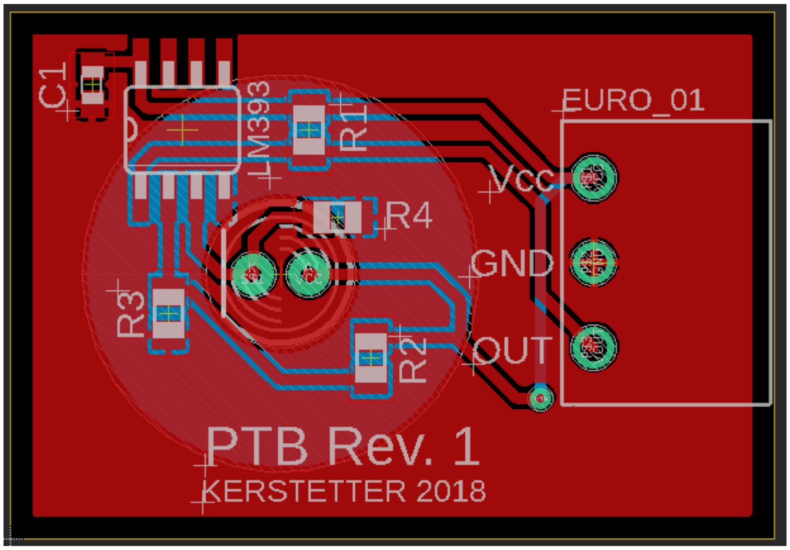

Figure 15

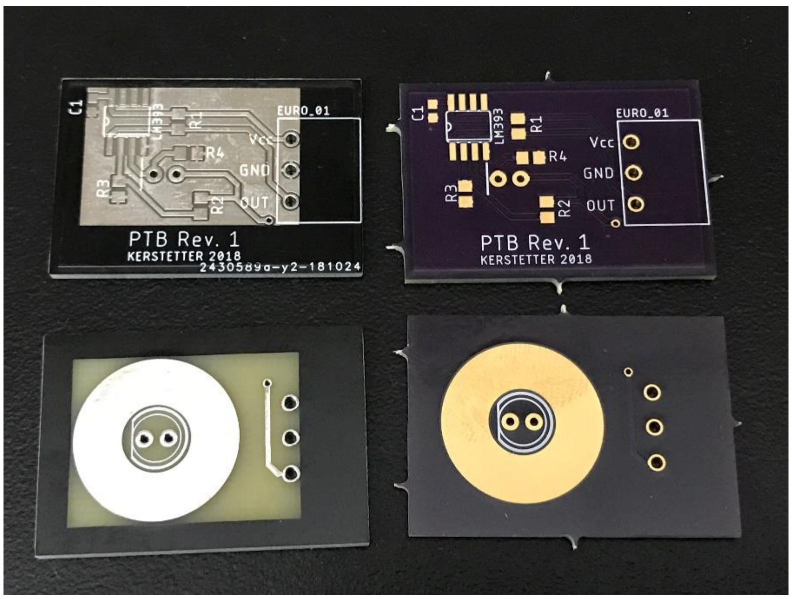

Figure 16 Fabricated PTBs (JLC PCB to the left and Osh Park to the right)

According, to the LM393 data sheet the IC can handle input voltages ranging from 2-36V. Therefore, the output voltage can be specified anywhere between 2-36V.

2.1.C. Complete Construction of the Digital Phototransistor Optoisolator

Entire DPO Circuit

Combining the LED Driver and the PTB creates the Digital Phototransistor Optoisolator. The circuitry of the entire DPO can be seen in figure 17.

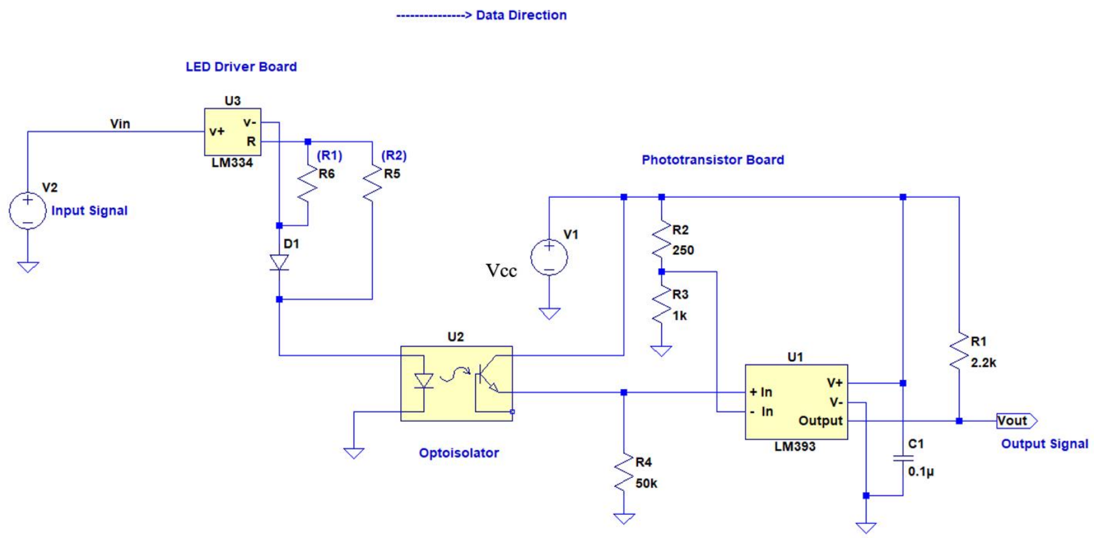

Figure 17

Test set values are set to simulate input and output waveforms of the DPO.

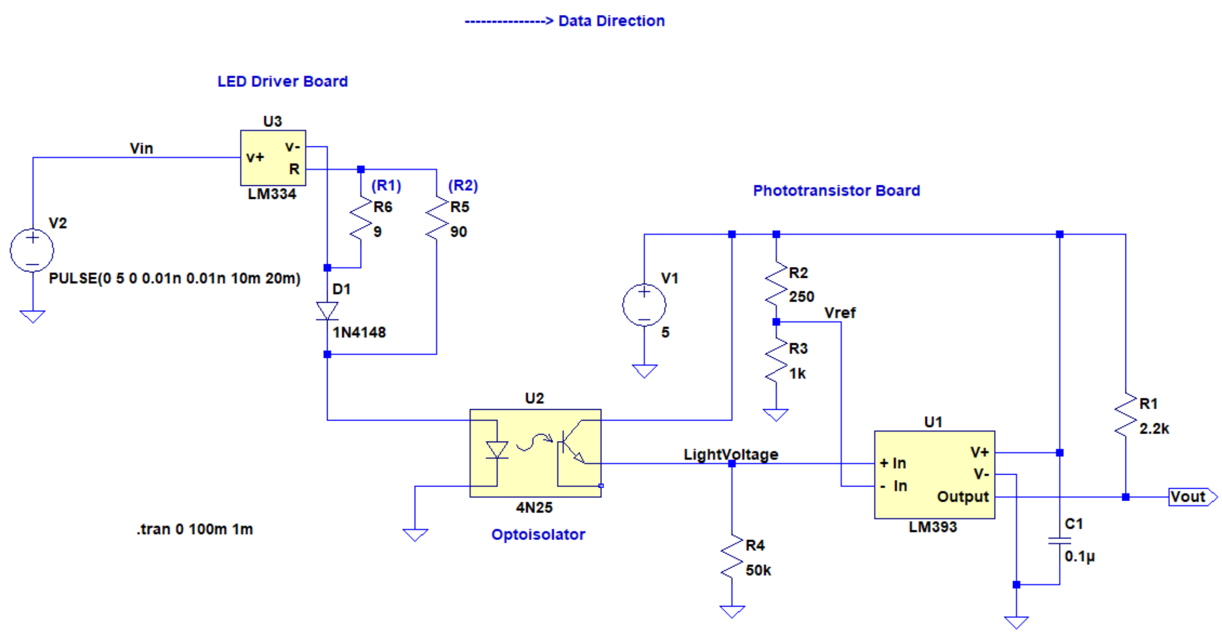

Figure 18

Figure 19 shows the input and output waveforms as specified by the parameters given in Figure 18. The frequency in the simulation results of figure 19 and 20 is 50 Hz.

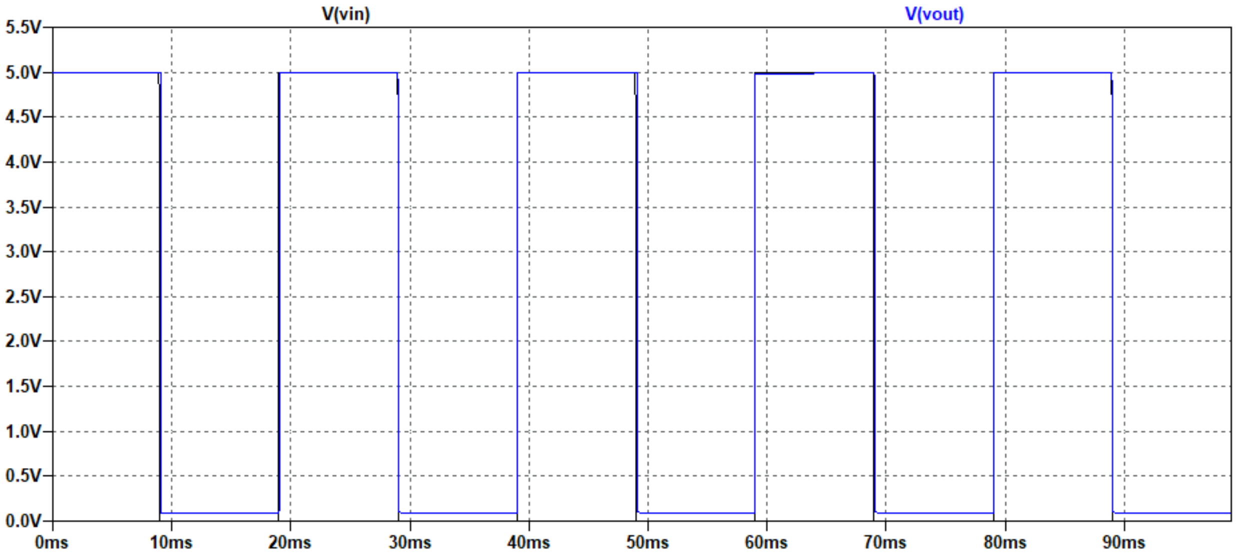

Figure 19 50 Hz

According to the simulation, the function of the DPO proves successful. The digital signal is successfully being sent from Vin to Vout. Figure 20 zooms in on the waveform as seen in Figure 19.

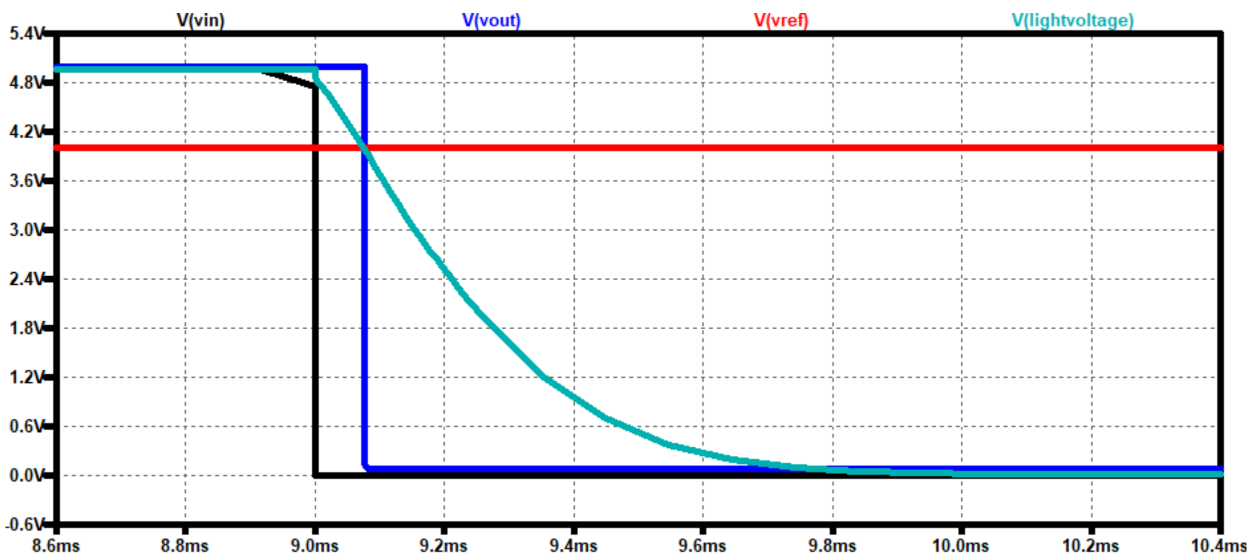

Figure 20 50 Hz

Now let's execute another simulation, but at a much greater frequency. Figure 21 shows an execution at 2.5kHz. There is 6u S delay in the input and output rising edges. While there is a 75uS delaying in the falling edges. This difference in delays causes pulse width distortion. This pulse width distortion will determine the bandwidth of the optoisolator itself.

The 6uS delay of the rising edges is due to the junction capacitance of the phototransistor. According to Mims, "the capacitor, formed by the junction, must be fully charged before the detector [phototransistor] can begin responding to all the pulse" (Mims). The easiest way to decrease latency in of the input and output rising edges is to decrease resistance value R4 (in figure 17 and 18). However, by decreasing this resistance the sensitivity of the photodetector decreases due to the fact that a phototransistor is a light-controlled current source.

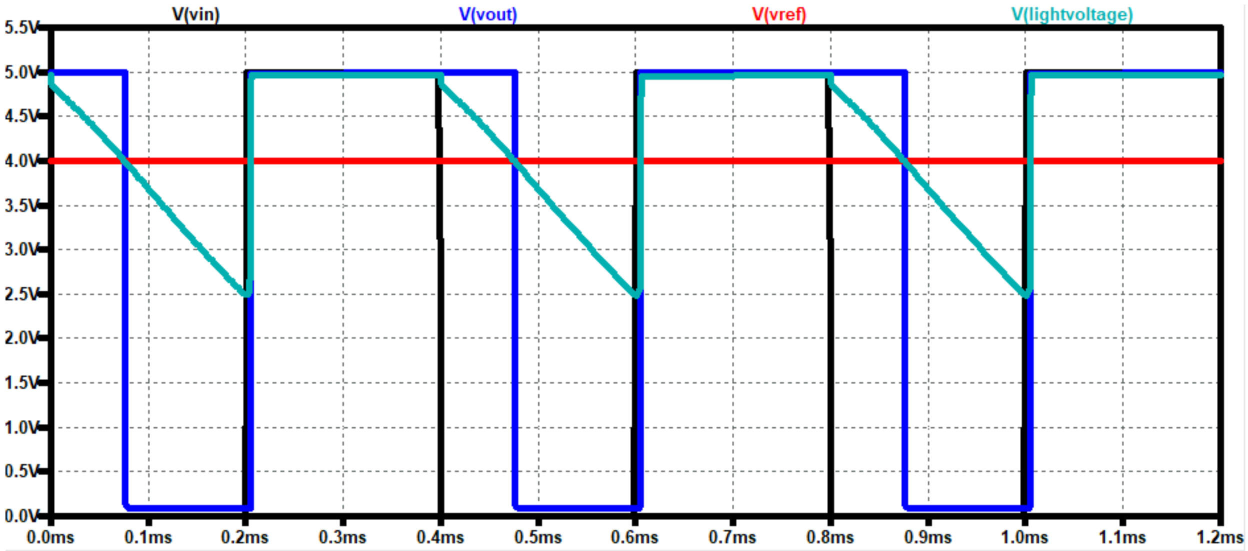

Figure 21 2.5kHz

If the frequency is too great the signal the signal does not pass successfully from Vin to Vout. A simulation where the frequency is 10kHz can be seen in figure 22.

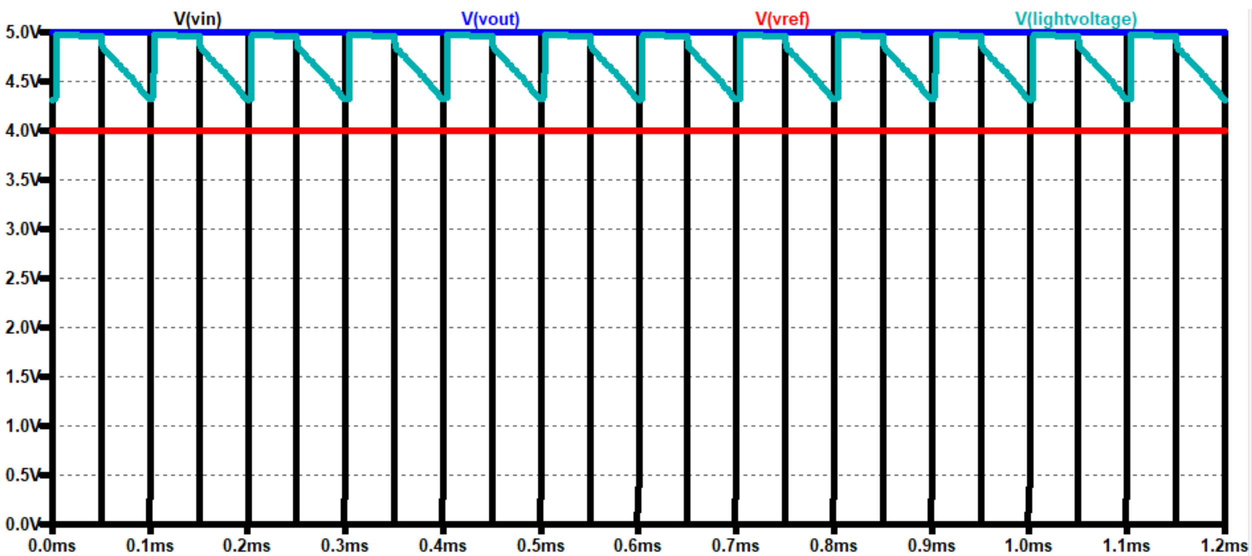

Figure 22 10kHz

The reason why the signal does not successfully transfer is because the decaying voltage *LightVoltage* never lowers below the reference voltage. The reason why *LineVoltage* is decaying is due capacitance in the LED itself. The capacitance in the LED causes the light intensity to decay at a certain rate. Therefore, the comparator will never output active low. The issue of pulse width distortion and frequency will be discussed in section 3.1.

First Prototype

The first prototype Digital Phototransistor Optoisolator was built as seen in figure 23. At this point the LED Driver PCB had been and designed and the PTB circuit was being designed. The proto board seen in the image contained an early version of the PTB circuit. The first prototype proved the DPO to function as intended and gave us the confidence to design a PCB for the PTB.

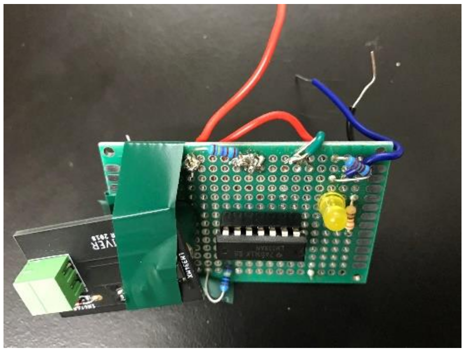

Figure 23

Selection of Phototransistor and LED

Initially, the first LEDs used, emitted light within the visible spectrum. This allowed us to be able to see the light signals being sent. The phototransistor that we chose to use was the Silicon NPN Phototransistor SFH 310 from OSRAM Opto Semiconductors. According to the given datasheet the relative spectral sensitivity is given in figure 24.

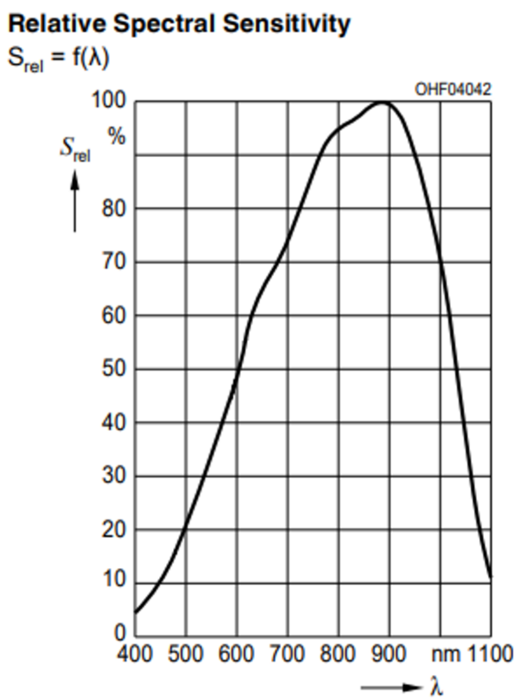

Figure 24

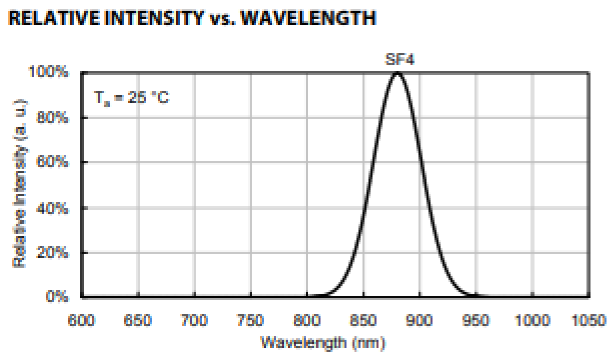

Figure 25

The phototransistor's peak spectral sensitivity is to light that has an 880nm wavelength. Light with an 880nm wavelength is considered infrared and is outside the visible spectrum. To maximize the sensitivity of the PTB itself an LED that emitted 880nm was needed. The LED we choose was the APT1608SF4C-PRV from Kingbright. In figure 25, a graph from the datasheet, shows that the chosen LED predominantly emits light with the wavelength of 880nm. The fine tuning of the resistance values (in figure 17 and 18) R2, R3, R4, R5, and R6 would allow the PTB to be particularly sensitive to light at or near 880nm.

Final Construction Form

Initially it was envisioned that the LED Driver and the PTB be connected to each other by a copper tube. However, copper being a conductor, some might argue that signals are being sent on the copper tube. To avoid any confusion, we decided to connect the LED Driver and the PTB together by an insulator. A connecting piece was designed and 3D printed as seen in figure 26 and 27. The connecting piece is glued to both boards.

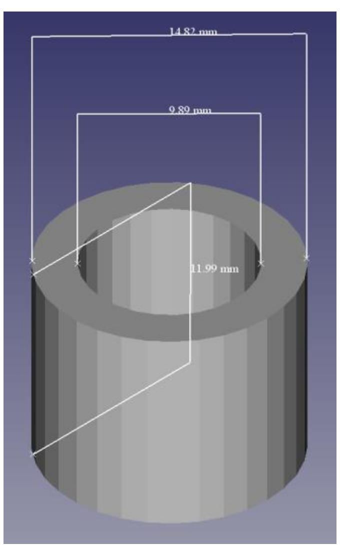

Figure 26

Figure 27

One DPO in its final construction form can be seen in figure 28 and 29.

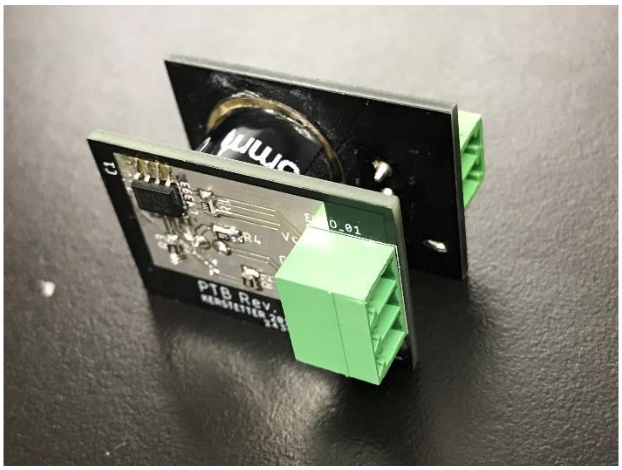

Figure 28 Prototype #2

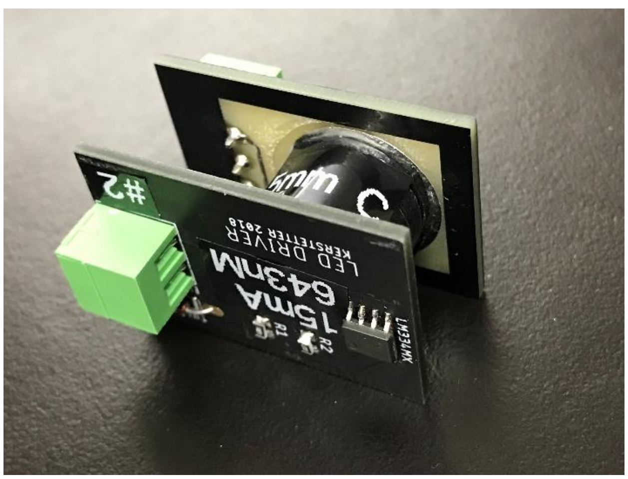

Figure 29 Prototype #2

Open Air Optical Channel

Initially, it was intended that the optical channel be enclosed and isolated. However, during the testing of the third prototype we realized that the communication between the two boards could occur at greater than intended distances. This lead us to use our third prototype for the Data Transfer System derivative project (see section 2.2.C).

DPO Level Shifting

The level shifting capability of the DPO is outlined in figure 30.

DPO Level Shifting		
	Min	Max
Input Voltage	~3.5	30
Specified Output Voltage	2	~10

Figure 30

Please note that the minimum input voltage varies with the resistance values (in figure 17 and 18) R2, R3, R4, R5, and R6. The maximum specified output voltage is also dependent upon these resistance values.

Resistances and Specialization

The resistance values (in figure 17 and 18) of R2, R3, R4, R5, and R6 determine three variables. The three variables are: reference voltage of the comparator, sensitivity of the photodetector, and light signal intensity. These three variables, if not properly set, will cause the DPO to be non-functioning. These resistances can be set such that the DPO is specialized for a specific purpose. Figure 31 details how these resistances effect the DPO.

Resistance	**Effect**
R2 & R3	Determines the Reference Voltage of the Comparator
R4	Determines the Sensitivity of the Photodetector
R5 & R6	Determines the Light Signal Intensity

Figure 31

These three variables can effect:

- Bandwidth
- Minimum Input Voltage
- Sensitivity
- Distance of Optical Channel (Transmission Distance)

Additional DPO Design Iterations
Prototype #2

Prototype #2 was the first prototype DPOs final construction form. This prototype used a different phototransistor. The phototransistor used was the HiLetgo B00M1PMH04 Phototransistor. This prototype also used an LED that emitted light with the wavelength of 643nm. The LED Driver was set such that 15mA would flow through the LED. Connection tube fully adhered to both boards. (Figures 28 and 29 contain photographs of prototype #2)

Resistance values used (as in figures 17 and 18):

- R1 = 2.211k Ω
- R2 = 9.76k Ω
- R3 = 9.76k Ω
- R4 = 880k Ω

Prototype #3

Prototype #3 was the first prototype DPO to contain an LED the emitted light outside the visible spectrum (880nm – infrared). This proved difficult to troubleshoot as it is impossible to ensure that the LED Driver is functioning with your naked eyes. The LED Driver was set such that ~15mA flowed through the LED. To ensure that the LED Driver is properly functioning. The phototransistor board can be used to detect the infrared light being emitted from the LED Driver. The LED that was used was APT1608SF4C-PRV from Kingbright and the phototransistor was Silicon NPN Phototransistor SFH 310 from OSRAM Opto Semiconductors. From this prototype onwards, all DPOs had this specific LED-phototransistor pair. Prototype, worked so well in free-space optical communication that this is the exact DPO used in the Data Transfer System (see 2.2.C).

Resistance values used (as in figures 17 and 18):

- R1 = 2.028k Ω
- R2 = 499 Ω
- R3 = 2.005k Ω
- R4 = 110k Ω

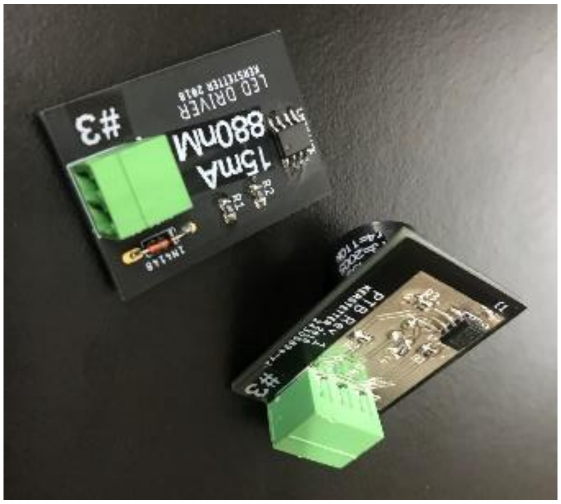

Figure 32 Prototype #3

The third prototype proved to be so successful that all subsequent DPOs were largely based upon Prototype #3.

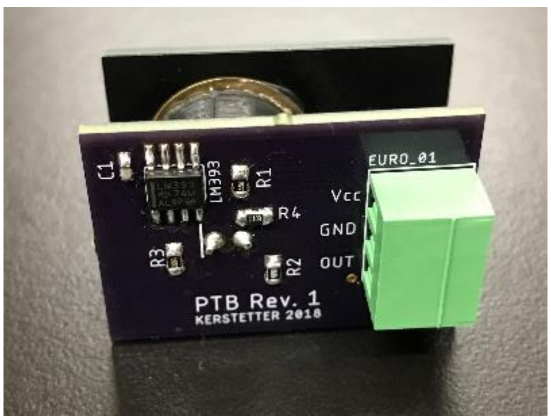

Figure 33 Subsequent DPO

Figure 34 Subsequent DPO

2.2. Applications

The Digital Phototransistor Optoisolator is an optoelectronic device that has an array of applications. The DPO is useful when electrical isolation is needed between circuitries. Additionally, the DPO can be used as a level shifter that has a great range of input and output levels. Three derivative projects have been developed to demonstrate some of the uses of the DPO.

2.2.A. Smart Switch

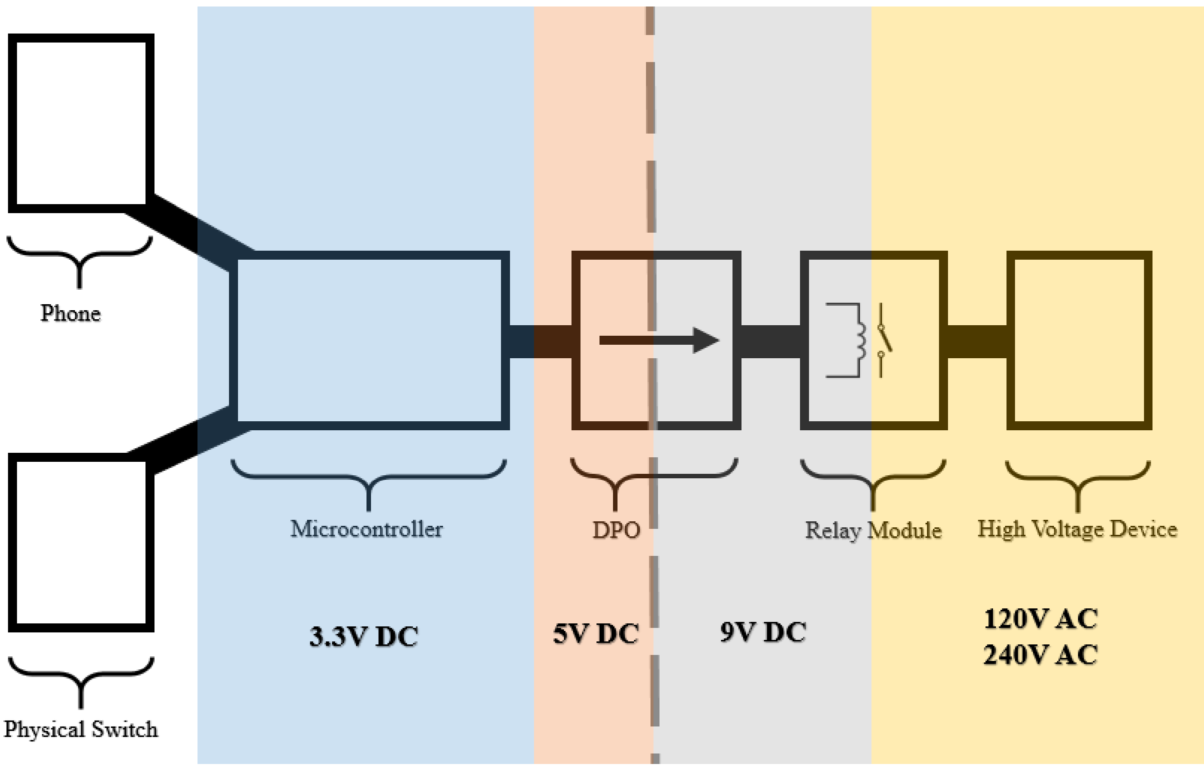

Figure 35

A common use of an optoisolator is to isolate low and high voltage systems. To demonstrate this application, we have created a Smart Switch (refer to figure 35). A microcontroller, the low voltage system, controls one DPO. The microcontroller outputs 3.3V DC. This signal is then shifted to 5V DC. To demonstrate the level shifting capabilities of the DPO we are using a relay module that requires 9V DC to be enabled. Therefore, the DPO has the input of 5V and a specified output of 9V. The relay module is the interface to 120V or 240V AC. If any in event that the relay module malfunctions; the microcontroller would be safe due to complete electrical isolation between the microcontroller and the relay module.

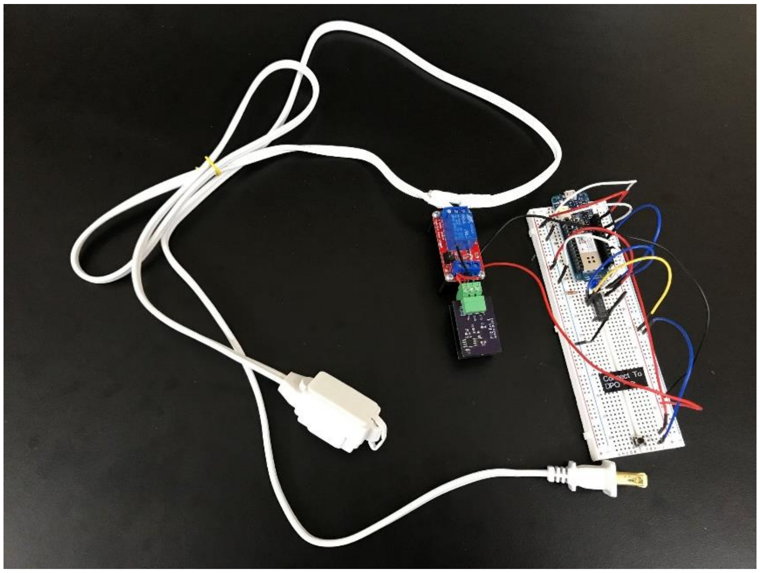

Figure 36 Smart Switch Implementation

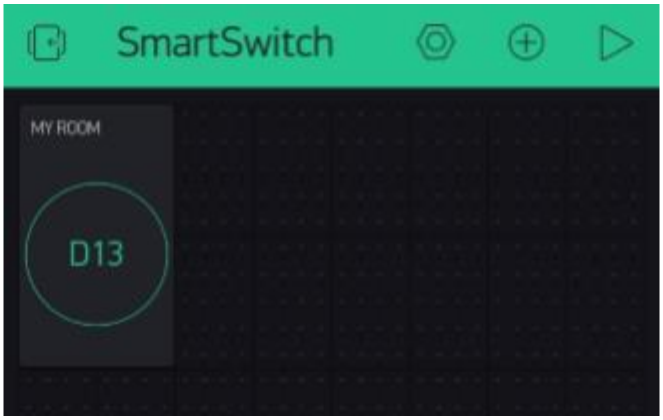

Figure 37 Blynk Interface

The smart switch has been designed such that the switch can be controlled in two methods. The first method is a physical button placed near the microcontroller. The high voltage device can be turned on and off by pressing the physical button. The second method is by a smart phone. In this method, a virtual button can turn on and off a high voltage device. Also, on the phone a timer can be set to turn on and off the high voltage device automatically.

The code that we used for the smart switch can be seen below:

```
#define BLYNK_PRINT SerialUSB
#include <SPI.h>
#include <WiFi101.h>
#include <BlynkSimpleWiFiShield101.h>
#include <Blynk.h>

char auth[] = "6a14f6d1ba3e4adfbfe715e2eed3f4f8";
char ssid[] = "AndroidAP";
char pass[] = "xwlm3507";

const int BUTTON_PIN = 11; //Our trusty button
```

```
const int OUT_PIN = 12;
const int LED_PIN = 10;
bool light = false;

int A;
int B;

BLYNK_WRITE(V13)
{

  SerialUSB.print("Got a value: ");
  SerialUSB.println(param.asStr());
}

void setup()
{
  // Debug console
  SerialUSB.begin(9600);
  pinMode(BUTTON_PIN,INPUT_PULLUP);
  pinMode(LED_PIN,OUTPUT);
  pinMode(OUT_PIN,OUTPUT);
  delay(100);
  digitalWrite(OUT_PIN,LOW);
  digitalWrite(LED_PIN,LOW);

  Blynk.begin(auth, ssid, pass);
}

void loop()
{

  delay(10);
  if(digitalRead(BUTTON_PIN)==LOW){
    while(digitalRead(BUTTON_PIN) ==LOW);
    light = !light;
  }

  if(light){
    digitalWrite(OUT_PIN,HIGH);
    delay(10);
  }
  else{
    digitalWrite(OUT_PIN,LOW);
    delay(10);
  }

  A = analogRead(A5);
  B = analogRead(A6);

  Serial.print("A = ");
  Serial.println(A);
  Serial.print("B = ");
  Serial.println(B);
  delay(100);

  if((A >=1022) || (B >= 900)){
    digitalWrite(LED_PIN,HIGH);
```

```
    delay(10);
  }
  else{
    digitalWrite(LED_PIN,LOW);
    delay(10);
  }

  Blynk.run();
}
```

2.2.B. Bidirectional Optical Link

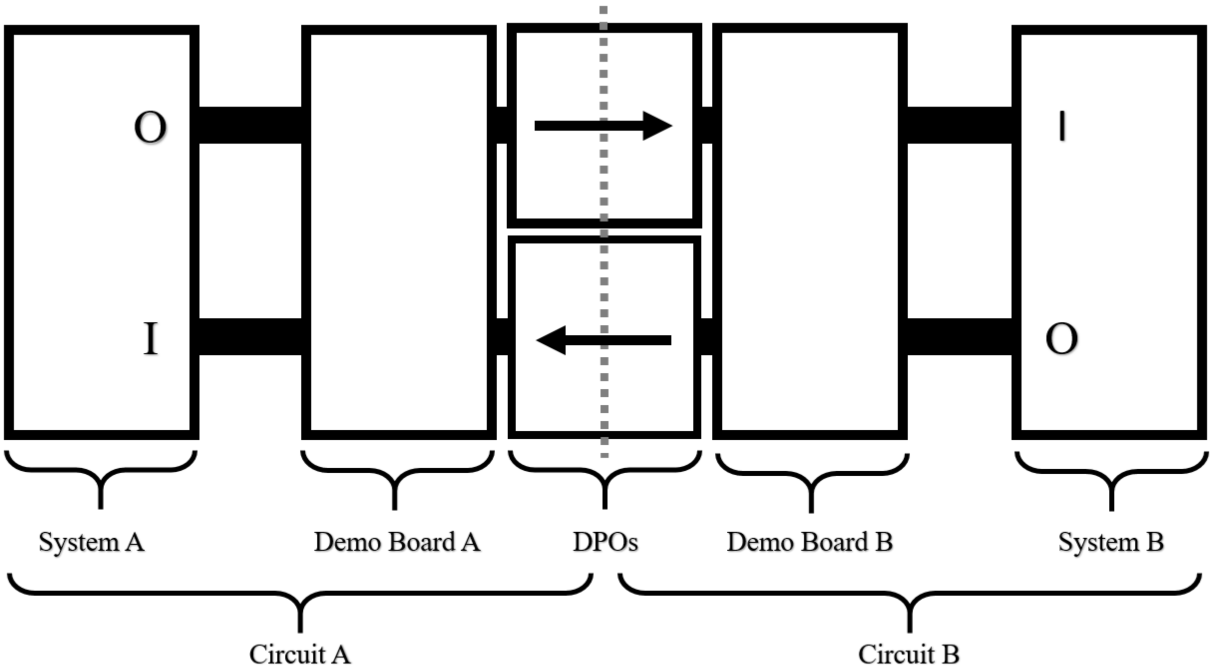

Figure 38

The Bidirectional Optical Link (BOL) allows for two systems to send digital signals to each other with complete electrical isolation. Figure 38 shows the breakdown of the implemented BOL. Two DPOs are the interconnection between system A and system B. The BOL contains 6 PCBs. The BOL requires two of the following: Demo Board, LED Driver, and the PTB.

There are two Demonstration (Demo) Boards that each power one LED Driver and one PTB. The circuit implemented on the Demo Board can be seen in figure 39. The PCB designed for the demo Board can be seen in figures 40 and 41. Off board circuitry was required to produce the correct circuitry as described in figure 39. A center-positive barrel jack was used to provide 9V or 12V DC to the board.

There are two different modes of the BOL. Mode 1 allows for a simple demonstration regarding the function of the BOL. A switch pushed on will light an indication LED on both Demo Board A and B. Mode 2 allows for external I/O from external Systems A & B.

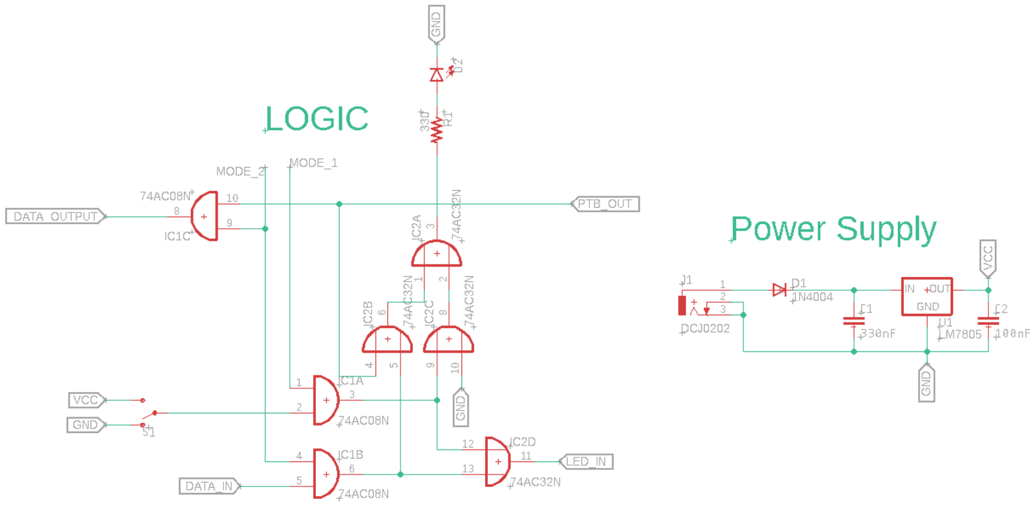

Figure 39

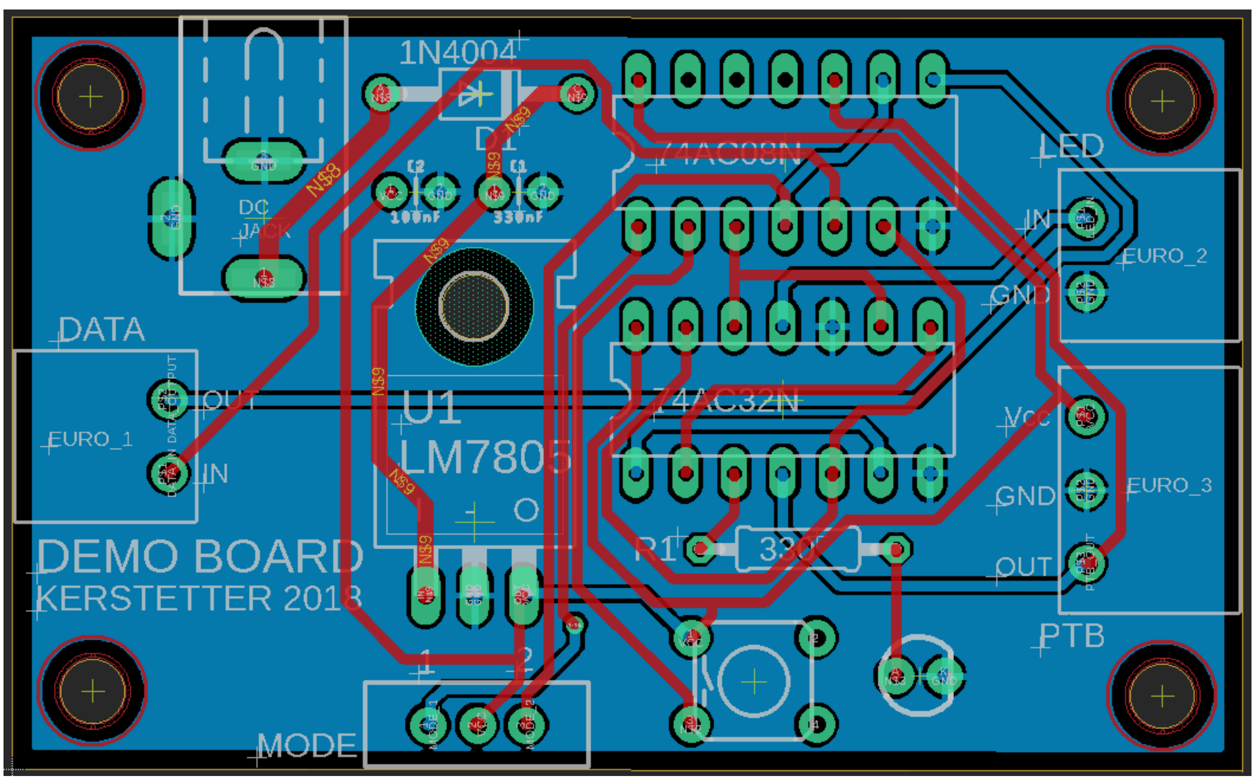

Figure 40

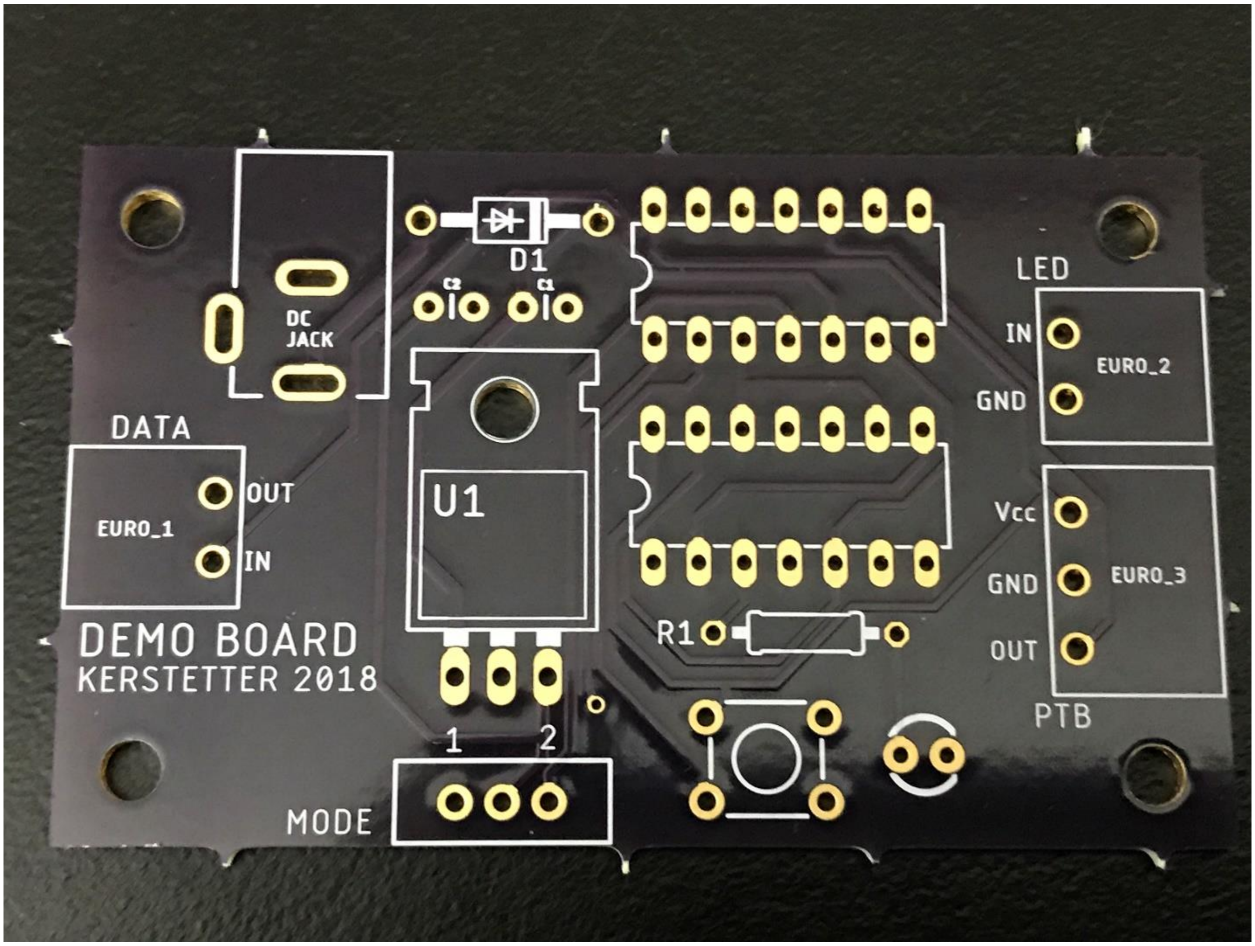

Figure 41

2.2.C. Data Transfer System

We have created a Data Transfer System (DTS) to demonstrate that DPOs can transfer data successfully. This system requires two microcontrollers. One microcontroller to take a specified 8-bit parallel input and send that data serially to a DPO. The receiving microcontroller receives the byte serially and displays the received byte in a parallel output. Figure 42 outlines the function of our DTS. Figure 43 contains a photo of the actual DTS implementation. The DPO was designed such that there was an enclosed and isolated optical channel. Having an open air channel where the LED Driver and the PTB are separated by a great distance (relative to the 13mm connection tube). The distance of separation is dependent upon proper alignment of boards and light interference. A generally reliable separation distance, in most external lighting conditions is 64mm. The separation distance can be greater, but proper data transfer is not guaranteed.

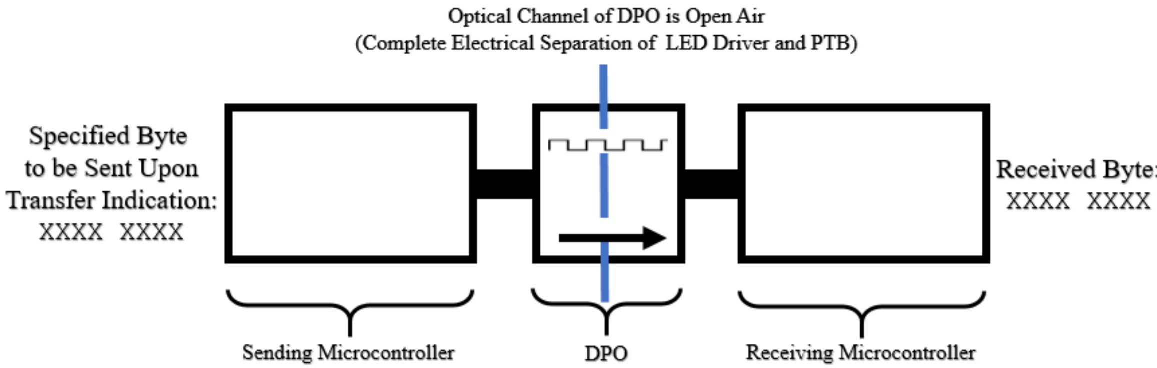

Figure 42

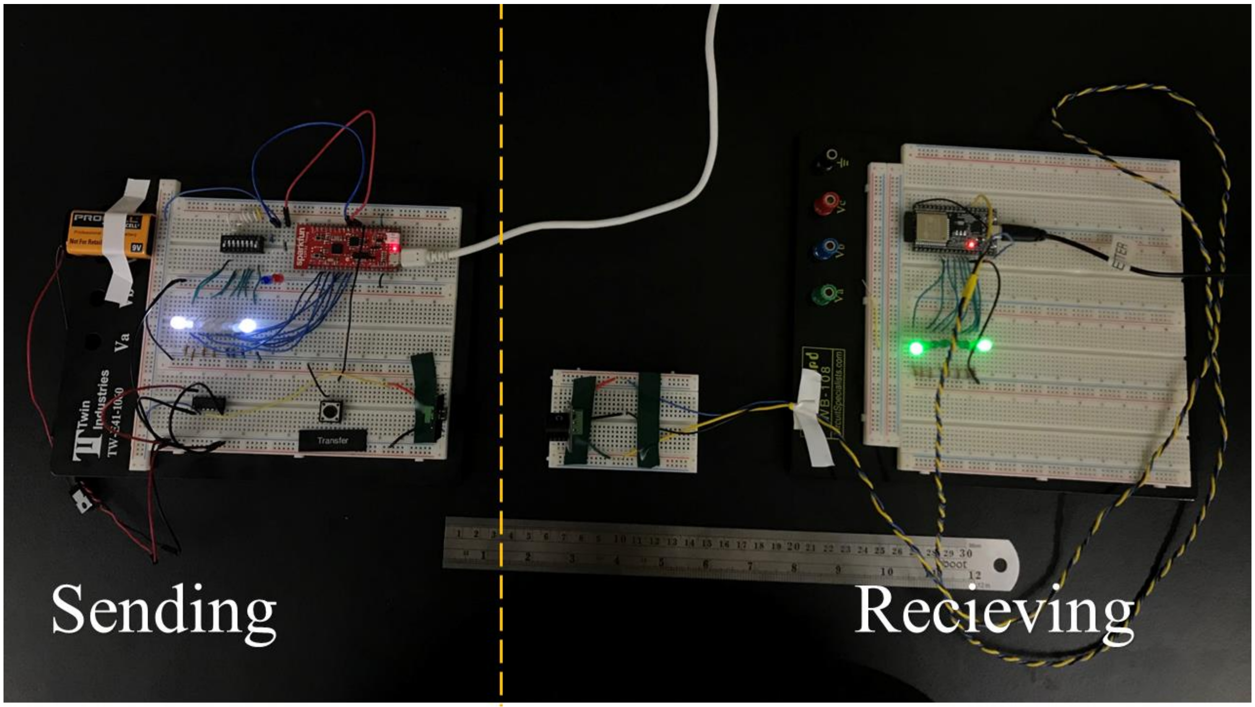

Figure 43

Figures 44 and 45 show signals from the sending microcontroller. A leading signal is sent to prime the receiving microcontroller. The receiving microcontroller is polling every 5ms for the leading signal. Once the leading signal is detected, the receiving microcontroller prepares to receive the data signal. The data signal is then sent at a rate of 33.33 Hz.

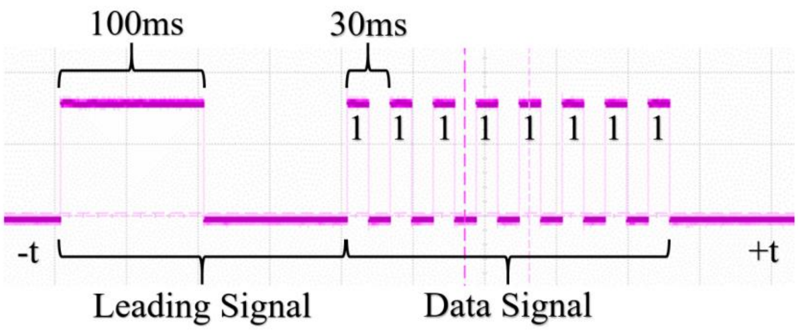

Figure 44

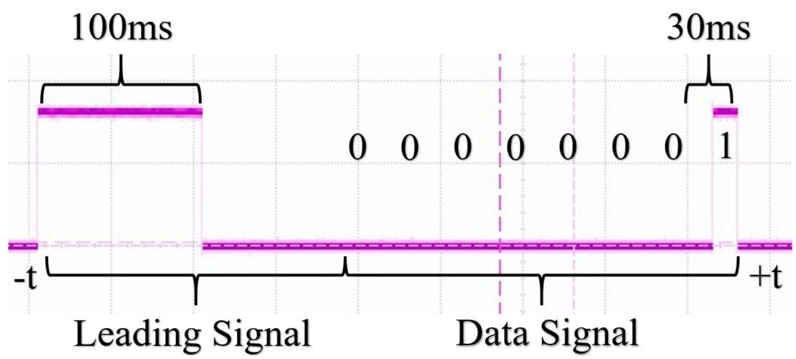

Figure 45

The following code is the code used by the sending microcontroller:

```
const int LED_PIN = 5; // The Sender LED
const int BUTTON_PIN = 0; //Our trusty button, located on the Board
const int LED2 = 2; //Button LED

/* int First through Eighth
 * are the individual Signals from the
 * parallel inputs
 */
int First;
int Second;
int Third;
int Fourth;
int Fifth;
int Sixth;
int Seventh;
int Eighth;

/* LDHigh is the time that the Sensor LED is on for. Represents bit 1.
 * LDLow is the time the LED will be off after the
 * LED goes high.
 * LDDelay is the High and LOW delays together, to represent bit 0.
 */
int LDHigh = 15; //15ms High
int LDLow = 15;  //15ms Low
int LDDelay = LDLow + LDHigh;
```

```
void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN,OUTPUT); //The sender LED output
  pinMode(BUTTON_PIN,INPUT_PULLUP); //Our button
  pinMode(LED2,OUTPUT); //Button LED
}

void loop() {
  if(digitalRead(BUTTON_PIN) ==LOW){
    while(digitalRead(BUTTON_PIN) ==LOW){
      digitalWrite(LED2,HIGH); //While Button is pressed
    };
    digitalWrite(LED2,LOW); //Turn off Button LED

    //BEGIN Reading Parallel Input
    First = analogRead(26);
    Second = analogRead(25);
    Third = analogRead(35);
    Fourth = analogRead(34);
    Fifth = analogRead(33);
    Sixth = analogRead(32);
    Seventh = analogRead(39);
    Eighth = analogRead(38);

    delay(10);

    //Print Parallel Inputs
    Serial.print("First: ");
    Serial.println(First);
    Serial.print("Second: ");
    Serial.println(Second);
    Serial.print("Third: ");
    Serial.println(Third);
    Serial.print("Fourth: ");
    Serial.println(Fourth);
    Serial.print("Fifth: ");
    Serial.println(Fifth);
    Serial.print("Sixth: ");
    Serial.println(Sixth);
    Serial.print("Seventh: ");
    Serial.println(Seventh);
    Serial.print("Eighth: ");
    Serial.println(Eighth);
    delay(10);

    //STARTUP Signal
    delay(10);
    digitalWrite(LED2,LOW);
    delay(10);
    digitalWrite(LED_PIN,HIGH); //STARTUP SIGNAL HIGH
    delay(100);
    digitalWrite(LED_PIN,LOW); //STARTUP SIGNAL LOW
    delay(100);

      //BEGIN SENDING SERIAL OUTPUT
      //IF LED is High, Send Signal, else, send NO signal
      if(First>100){
```

```
    digitalWrite(LED_PIN,HIGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
    delay(LDLow);
  }
  else
    delay(LDDelay);

  if(Second>100){
    digitalWrite(LED_PIN,HIGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
    delay(LDLow);
  }
  else
    delay(LDDelay);

  if(Third>100){
    digitalWrite(LED_PIN,HIGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
    delay(LDLow);
  }
  else
    delay(LDDelay);

  if(Fourth>100){
    digitalWrite(LED_PIN,H IGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
    delay(LDLow);
  }
  else
    delay(LDDelay);

  if(Fifth>100){
    digitalWrite(LED_PIN,HIGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
    delay(LDLow);
  }
  else
    delay(LDDelay);

  if(Sixth>100){
    digitalWrite(LED_PIN,HIGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
    delay(LDLow);
  }
  else
    delay(LDDelay);

  if(Seventh>100){
    digitalWrite(LED_PIN,HIGH);
    delay(LDHigh);
    digitalWrite(LED_PIN,LOW);
```

```
        delay(LDLow);
      }
      else
        delay(LDDelay);

      if(Eighth>100){
        digitalWrite(LED_PIN,HIGH);
        delay(LDHigh);
        digitalWrite(LED_PIN,LOW);
        delay(LDLow);
      }
      else
        delay(LDDelay);
      //END
    }
  delay(50); //Delay added to keep the residue from the Button LED away

  Serial.println("--------------------------");
  Serial.println();
 }
```

The following code is the code used by the receiving microcontroller:

```
#define LED1 13
#define LED2 12
#define LED3 14
#define LED4 27
#define LED5 26
#define LED6 25
#define LED7 33
#define LED8 32
/* These are all of the LEDs that we will be using to OUTPUT
 * the received serial signal
 */

int dataIn = 4; //This is our Serial input
int dataRead1=0;
int dataRead2=0;
int dataRead3=0;
int dataRead4=0;
int dataRead5=0;
int dataRead6=0;
int dataRead7=0;
int dataRead8=0;
/*  dataRead1 thru 8 will be reading the individual Received signals.
 */


void setup() {
  Serial.begin(115200); //Setup the Serial monitor

  pinMode(LED1,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);
  pinMode(LED4,OUTPUT);
  pinMode(LED5,OUTPUT);
```

```
  pinMode(LED6,OUTPUT);
  pinMode(LED7,OUTPUT);
  pinMode(LED8,OUTPUT);
  /* LED1 thru 8 are setup as OUTPUTS and these will help turn ON
   * the individual LEDs
   */
}

void loop() {
  int dataRead0; //This will catch the startup signal from Receiver
  int myDelay = 30; //The interval between each signal

  delay(5); //Polling every 5ms

  dataRead0 = analogRead(dataIn); //Reading till we receive a Startup Signal
  Serial.print("Reading: ");
  Serial.println(dataRead0);
  Serial.println();
  /* This is where the Receiver will start to read the individual signals
   * after a successful reading of the Startup Signal
  */

  if(dataRead0>=3000){
    delay(200); //Delay set for the Startup Signal, Reading
    //First bit
    dataRead1 = analogRead(dataIn);

    //Second
    delay(myDelay);
    dataRead2 = analogRead(dataIn);

    //Third
    delay(myDelay);
    dataRead3 = analogRead(dataIn);

    //Fourth
    delay(myDelay);
    dataRead4 = analogRead(dataIn);

    //Fifth
    delay(myDelay);
    dataRead5 = analogRead(dataIn);

    //Sixth
    delay(myDelay);
    dataRead6 = analogRead(dataIn);

    //Seventh
    delay(myDelay);
    dataRead7 = analogRead(dataIn);

    //Eighth
    delay(myDelay);
    dataRead8 = analogRead(dataIn);


    delay(100); //End delay to keep out the residue from last signal
```

```
    //Print the Values
    Serial.println("Values: ");
    Serial.print("LED1 = ");
    Serial.println(dataRead1);
    Serial.print("LED2 = ");
    Serial.println(dataRead2);
    Serial.print("LED3 = ");
    Serial.println(dataRead3);
    Serial.print("LED4 = ");
    Serial.println(dataRead4);
    Serial.print("LED5 = ");
    Serial.println(dataRead5);
    Serial.print("LED6 = ");
    Serial.println(dataRead6);
    Serial.print("LED7 = ");
    Serial.println(dataRead7);
    Serial.print("LED8 = ");
    Serial.println(dataRead8);

    Serial.println("----------------------");
  }
  delay(10);

  /* This is the part of the code where
   * we light up the individual LEDs
   * based off the received signals
   */
  if(dataRead1>=3000)
    digitalWrite(LED1,HIGH);
  else
    digitalWrite(LED1,LOW);

  if(dataRead2>=3000)
    digitalWrite(LED2,HIGH);
  else
    digitalWrite(LED2,LOW);

  if(dataRead3>=3000)
    digitalWrite(LED3,HIGH);
  else
    digitalWrite(LED3,LOW);

  if(dataRead4>=3000)
    digitalWrite(LED4,HIGH);
  else
    digitalWrite(LED4,LOW);

  if(dataRead5>=3000)
    digitalWrite(LED5,HIGH);
  else
    digitalWrite(LED5,LOW);

  if(dataRead6>=3000)
    digitalWrite(LED6,HIGH);
  else
    digitalWrite(LED6,LOW);
```

```
  if(dataRead7>=3000)
    digitalWrite(LED7,HIGH);
  else
    digitalWrite(LED7,LOW);

  if(dataRead8>=3000)
    digitalWrite(LED8,HIGH);
  else
    digitalWrite(LED8,LOW);
}
```

2.3 Description of Roles

Bryan Kerstetter

- Conception of Project Topic
- Design of LED Driver
- Design of Phototransistor Board (PTB)
- Design of Demonstration Board (Demo Board)
- Design of Digital Phototransistor Optoisolator
- Design of Bidirectional Optical Link
- Design of Smart Switch
- Design of Data Transfer System
- Troubleshooting
- All soldering work
- Presentation
- Final Report

David Santiago

- Design of Data Transfer System
- Programming for Data Transfer System
- Programming for Smart Switch
- Frontend electronics for microcontrollers
- Microcontroller Troubleshooting
- Troubleshooting
- Presentation

3. Encountered Problems

3.1. Pulse Width Distortion and Frequency

Under most configurations the DPO could successfully operate at 10 kHz. As noted previously, the pulse width distortion of the input and the output signals determine the frequency of the signal that is able to successfully pass through the DPO. The root of the pulse width distortion is believed to be the capacitance in the LED as discussed in section 2.1.C. The pulse width distortion can be minimized by increasing the reference voltage of the comparator. By increasing the reference voltage of the comparator the output falling edge occurs sooner. Figure 46 is an adapted version of figure 21. In Figure 46 there are two reference voltages: A at 4V and B at

4.75V. Reference voltage A is the reference voltage used in the simulation portion of section 2.1.C. A leads to a falling edges delay of 75uS. Whereas, B leads to a 10uS delay. Thereby, minimizing the pulse width distortion factor at this frequency and allowing a greater frequency pass through the DPO.

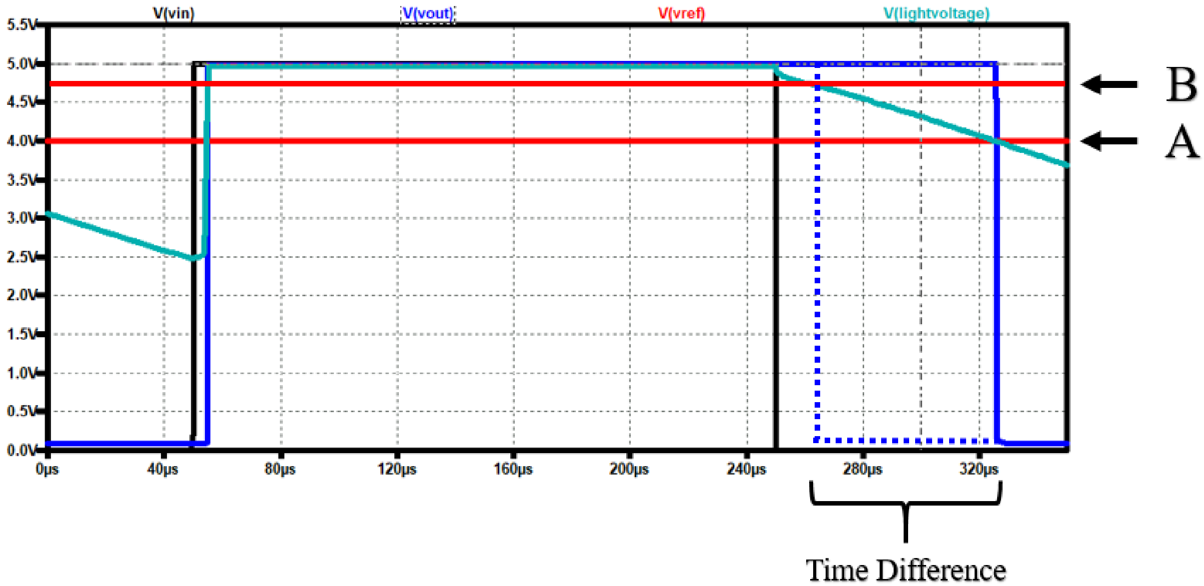

Figure 46 2.5kHz

Figure 22 demonstrated a situation where the DPO was unable to successfully operate with a frequency of 10kHz. However, when the reference voltage is raised from A to B; the DPO is able to operate at 10kHz as demonstrated in figure 47.

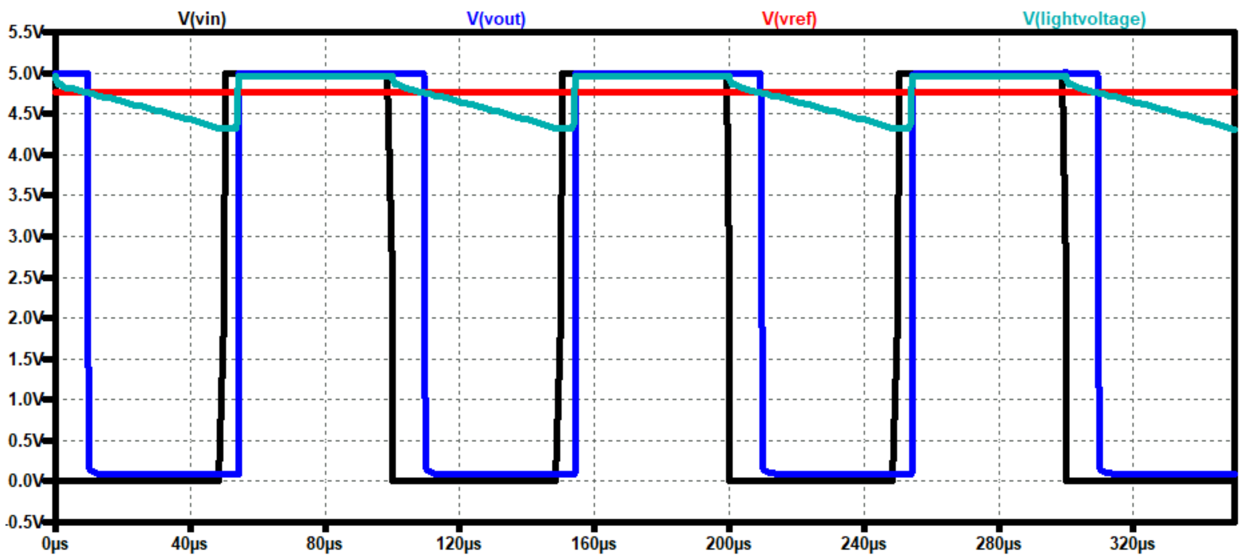

Figure 47

In The Future

A higher speed LED and phototransistor could be used to allow the DPO to operate at higher frequency. The circuit could PTB circuit could be completely redesigned such that the load resistance to the phototransistor is less.

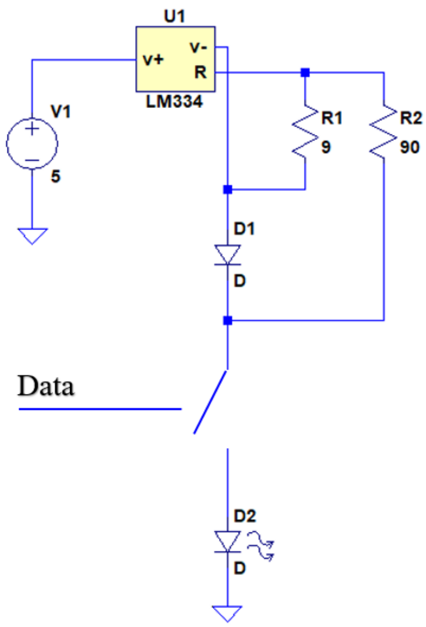

Figure 48 (compare to figure 4 or 8)

However, we believe the biggest bandwidth bottle neck on the DPO was the LM334s relation to our circuit. The LM334 was not designed to be powered on and off at rapid rates. Therefore, a switching circuit could be implemented that would allow the DPO to hand greater bandwidths (refer to figure 48).

3.2. Optical Interference and Transmission Distance

Additional problems were encountered during the Data Transfer System derivative project. In an open-air free-space optical channel, as demonstrated and used, there were issues of optical interference. Light from other sources reached the PTB and lead to inaccurate data transfer. This was reduced by super gluing a connection tube to the PTB as seen in figure 49. Additionally, the DPO used in the data transfer system was configured such that it was especially sensitive to the light emitted from the LED used. However, optical interference is still an issue.

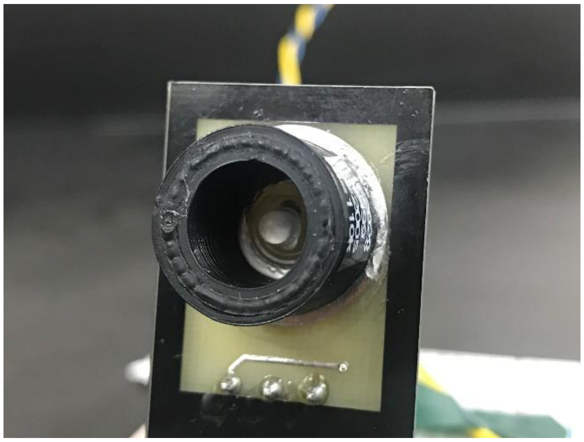

Figure 49

Another issue encountered was the issue of increasing the separation of the LED Driver and the PTB such the optical channel and transmission distance is greater. Experimentally, a reliable transmission distance was 64mm.

In the future, we would like to possibly use parabolic reflectors and/or lasers to increase the transmission distance.

4. Summary

In summary, a DPO can be built with discrete components on two boards. The optoisolator consists of sending and receiving boards with an optical channel connecting them. The DPO can be built such that it has level shifting capabilities while providing complete electrical isolation.

5. Conclusions

Our project goals were to create an optoisolator that could send signals across an optical channel and to provide a means of complete electrical isolation between circuits. These goals were achieved. Additionally, our optoisolator is capable of level shifting and free-space optical communication. This project gave us an introduction to the exciting world of optoelectronics. In the future, we would like to take what we have learned and work on creating an optoisolator that can operate at higher frequencies. While, reducing optical interference and increasing transmission distance.

Note: The LTspice model of the DPO does not perfectly represent the DPO itself. The specific LED and phototransistor used could not be implemented into the LTspice model. However, the LTspice model was helpful throughout the design process.

Acknowledgments

We would like to thank Angsuman Roy for advice regarding the LED Driver. Also, we would like to thank Daniel Senda for modeling the connection tube. Additionally, we would like to thank Vikas Vinayaka for valuable input. Finally, we would like to thank Dr. Grzegorz Chmaj for lending us the microcontrollers used throughout this project.

References:

Kang, Sae-Kyoung. “Demonstration of Bidirectional Optical Link on Optical PCB Using a Driver-Receiver Combined CMOS Transceiver .” *IEEE*, 2006, ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1645867&tag=1.

“LM134/LM234/LM334 3-Terminal Adjustable Current Sources.” Texas Instruments, May 2013.

“LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators.” Texas Instruments, Oct. 2018.

Mims, Forrest M. *LED Circuits & Projects*. Howard W. Sams & Co., Inc., 1973.

“NPN Phototransistor SFH 310 .” OSRAM Opto Semiconductors.

Sedra, Adel S., and Kenneth C. Smith. *Microelectronic Circuits*. Oxford University Press, 2016.