9-Week Project - Documentation - Laser Light Communication System

In this Documentation:

General Overview

Abstract

The goal of this project is to have a user type characters on a keyboard at one end and have them displayed on the other via a separate display. An Arduino Due, which is a microcontroller (similar to, but less powerful than, a little computer), that takes input from the keyboard, processes that input, and sends an output to its display, LED, and the laser as prompted. The laser then transmits the data to the other side by flashing light pulses (similar to fiber optics, but without the cable). The light sensor on the other side picks up this transmitted signal, which is interpreted by an Arduino Uno, another microcontroller. The Arduino Uno then processes that input and sends text to a display or turns a white LED on and off, which is used for those who know Morse Code.

User's Guide

In this Guide:

Setup

- Turn on hallway circuit breaker 2 to route power to the units.
- Make sure the switch on each unit's board is flipped to the upward position. This powers each unit on.
- Make sure the laser is aligned before operation.

Operation

General

- Press any character key to send it to the display.
- Backspace, tab (3-space), delete, arrows keys (excluding up), and enter can be used within the character display.
    - *Note: words are not shifted to the left with a deleted character.*
    - *Note: pressing backspace while the cursor is in the top-left will result in the cursor at the bottom-right.*

Special Keys

- **HOME Key:** Pressing the HOME key clears the screen and sets the cursor at the top-left position.
- **END Key:** Holding the END key will result in the white LED turning on (both ends). Releasing the END key turns the LED off. (Used for Morse Code)
- **Escape Key (ESC):** Pressing the ESC key will turn the display screen on or off.

Alignment

- **F1 Key:** Press the F1 key toggle the laser on or off. This is used for alignment.
- The laser mount has adjustable windage and elevation knobs, which can be used to adjust the laser's direction.
- Adjustments can also be done by shifting the photodiode (or laser) assembly slightly, but do not rely on assembly movement as the main means of alignment.

Shutdown

- When done using the system, please ensure the switches on both units (one each) are flipped in the down ("off") position.

Devices

Light Receiver

- Small device on receiving end mounted on a DIN rail, on the South side of its I-beam.

Laser

- Mounted in a scope-mount assembly on transmitting end, on the West side of its angled-beam.

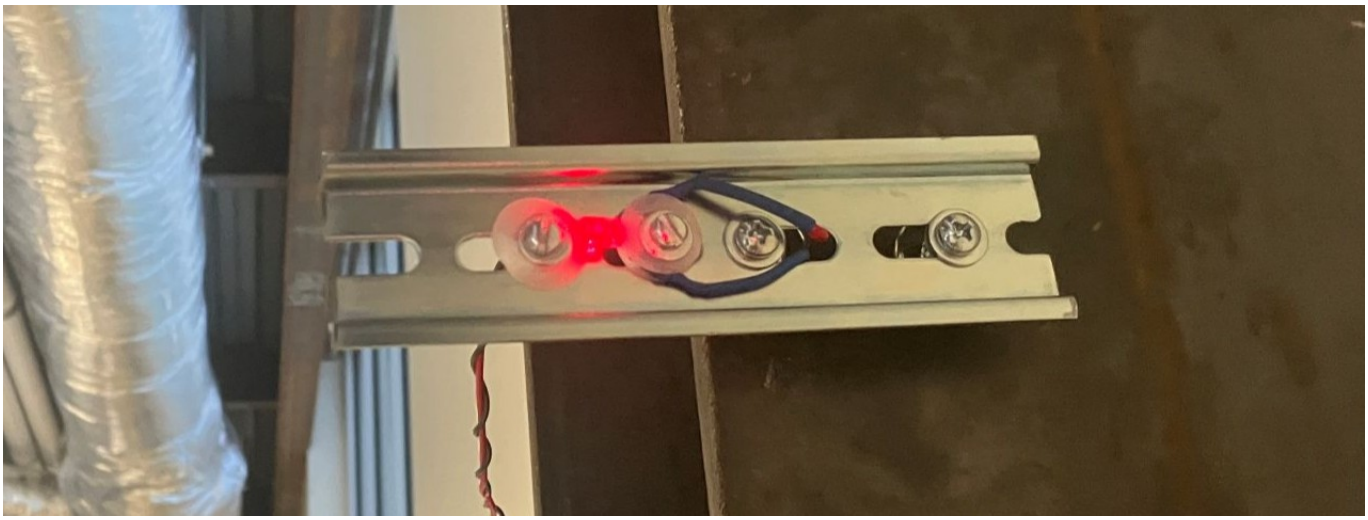
Project Pictures

Displays


This is a sentence.
It id being displaye
d as I type. Showing
this for a test._

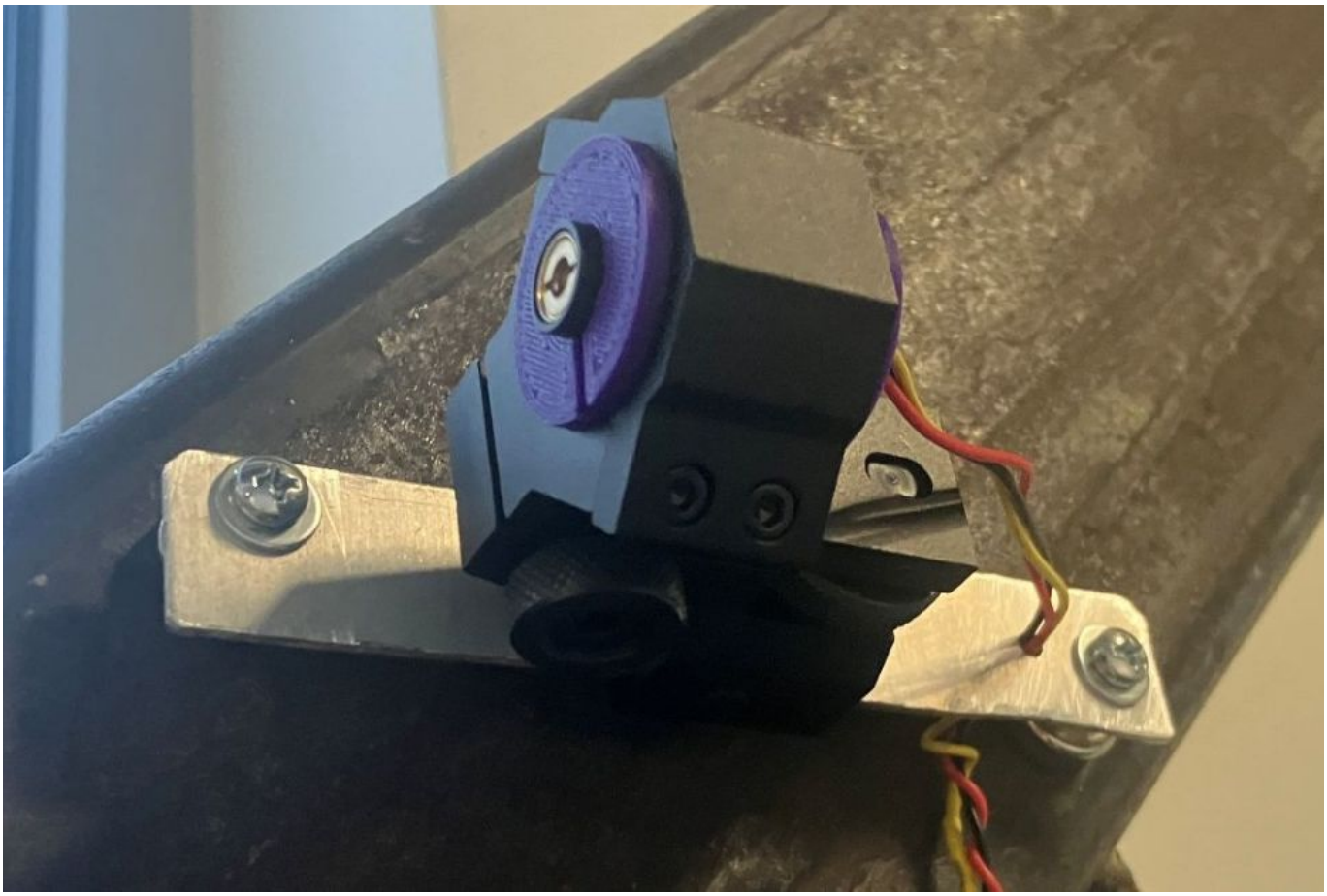
- Ignore the misspelling of the word "is" displayed as "id".

Laser Beam Active



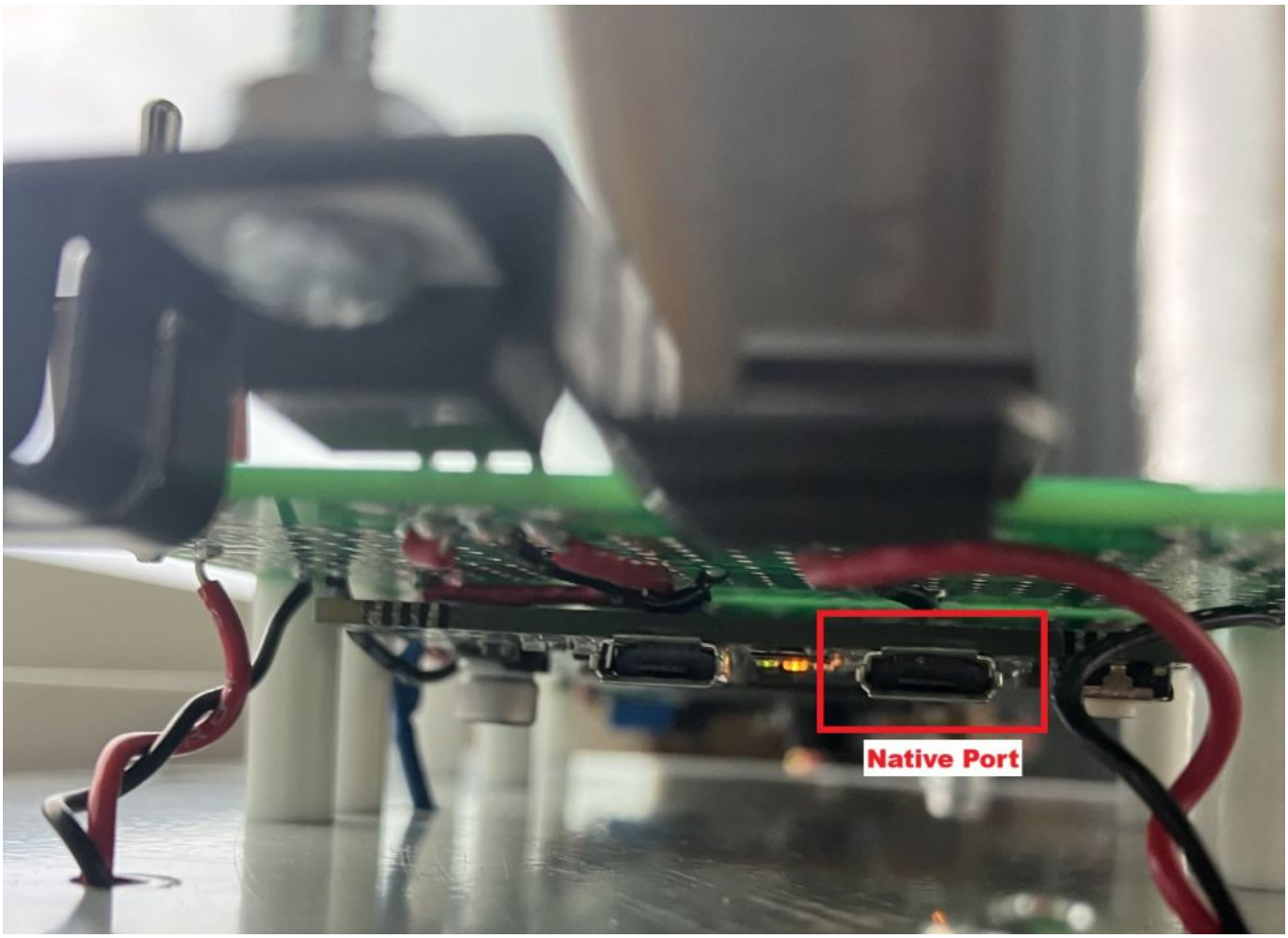
- Both of these pictures were taken with the laser in alignment mode (laser constantly on).

Laser Beam In-active



Keyboard Connection Procedure

- If the keyboard cord happens to become disconnect, you will notice that there are two different ports on the underside of the transmitting unit:



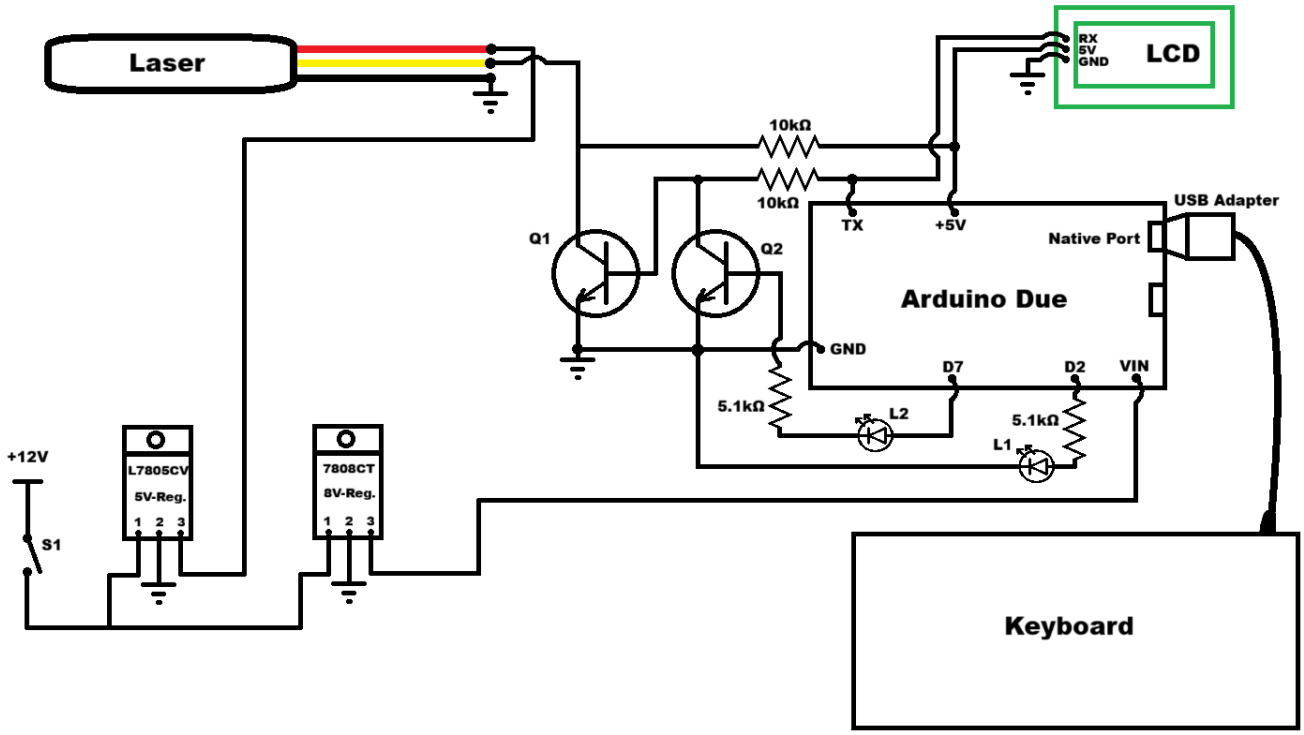
- The port on the right side (far edge of the board) is the native port, while the port on the left (middle of the board) is the programming port. The USB adapter needs to be connected to the native port (on the right).
- If after connecting the USB and the keyboard still doesn't result in characters on the display, try press CAPS-LOCK, NUMLOCK, or SCROLL-LOCK and look for a little green light on the corresponding indicator. If you see a green light, the keyboard is connected, and resetting the power to the board may fix the problem. If the lights don't turn on, then the USB is likely not connected. Try wiggling the USB around and get it to fit in its port. If still nothing occurs, try resetting the power.
- Please be gentle when inserting the USB adapter into its port.
- **Note:** The adpater will be at a slight angle when insert into the port. This is not a problem, but just a minor design flaw. The amount that it is angled is barely noticeable, but there is no reason to worry about it.

Project Details

Documentation Overview

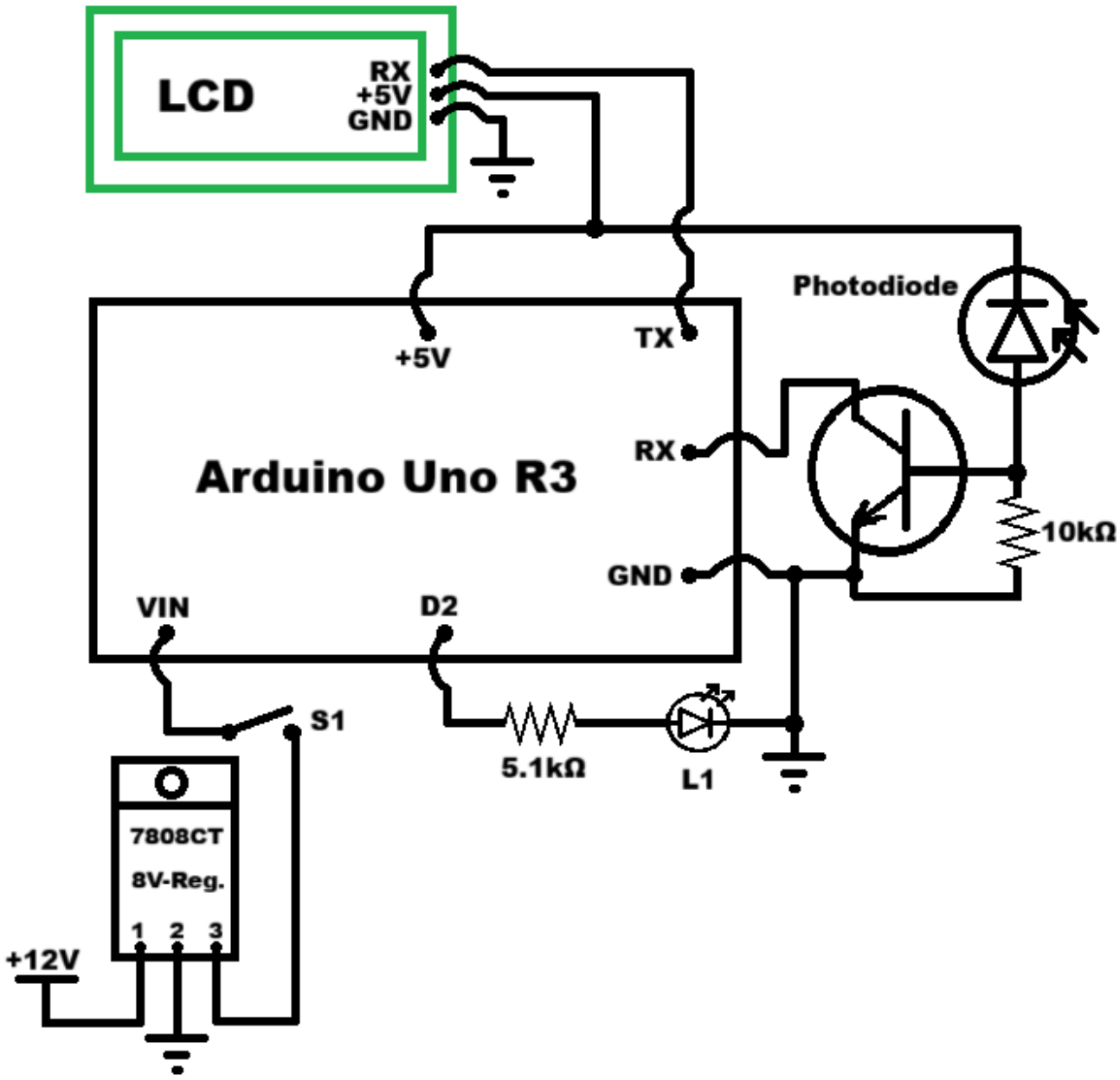
Final Schematics

Arduino Due (Transmitter)



- Note how LED L2 is connected in series with a 5k1 resistor, digital pin 7 (D7), and transistor Q2's base terminal. This meant to serve as an indicator that the laser is supposed to be held on. However, if this LED ever fails open, it will result in the alignment function not working, and the laser not being able to be held on.

Arduino Uno R3 (Receiver)



Final Project Code

- Arduino IDE was used to write, compile, and upload the code.

Final Arduino Due Code

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0; //Used to store value of detected normal key.
int OemKey = 0; //Used to store value of detected special key.
int LED = 2;
```

```
int alignPin = 7; //Used for aligning the laser.
const int morseKey = 77;

boolean screenPower = true;
boolean laserTest = false; //Testing alignment, yes/no.

void setup(){
  pinMode(LED, OUTPUT);
  pinMode(alignPin, OUTPUT);
  digitalWrite(LED, LOW);
  digitalWrite(alignPin, LOW);
  laserTest = false;

  Serial.begin(9600);
  delay(10);
  Serial.write(12); //Clear screen.
  delay(10);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  OemKey = keyboard.getOemKey(); //Assign the Oem key to "OemKey".
  if (key == 0) { //A standard key is a non-zero value, while a special key
registers as a zero-value.
    sendCommand(OemKey); //Process special key.
  } else {
    Serial.write(key); //Send a normal pressed key.
    if (OemKey == 40 || OemKey == 88) { //Enter Keys
      Serial.write(13); //Return to next line.
    }
  }
}

void keyReleased() {
  OemKey = keyboard.getOemKey();
  if (OemKey == morseKey) {
    Serial.write(15); //Used to turn other LED off.
    digitalWrite(LED, LOW);
  }
}


void sendCommand(int code) {
  switch (code) {

    case 0:
      break;

    //Morse Code Key
    case morseKey:
```

```
      Serial.write(14);
      digitalWrite(LED, HIGH);
      break;

    //Arrow Keys
    case 81: //Down
      Serial.write(10); //Cursor down.
      break;
    case 80: //Left
      Serial.write(8); //Cursor left.
      break;
    case 79: //Right
      Serial.write(9); //Cursor right.
      break;

    //Cluster
    case 76: //Delete
      //Just replace with space.
      Serial.write(" ");
      Serial.write(8);
      break;
    case 74: //Home
      Serial.write(12); //Refresh Display ("Form Feed")
      break;

    //Misc
    case 42: //Backspace
      //Functions like backspace.
      Serial.write(8);
      Serial.write(" ");
      Serial.write(8);
      break;
    case 43: //Tab
      Serial.print("   "); //Three spaces.
      break;
    case 41: //Escape
      powerLCD();
      break;
    case 58: //F1
      toggleLaser(); //Turn laser on/off for alignment.
      break;

    //Default
    default:
      break;
  }
}

//Turns LCD on or off.
void powerLCD() {
  if (screenPower) {
    Serial.write(21); //Turns LCD screen off. Doesn't erase characters.
    screenPower = false;
  } else {
```

```
    Serial.write(24); //Turns LCD screen on.
    screenPower = true;
  }
}

//Tuns laser on/off for alignment.
void toggleLaser() {
  if (laserTest) {
    digitalWrite(alignPin, LOW);
    laserTest = false;
  } else {
    digitalWrite(alignPin, HIGH);
    laserTest = true;
  }
}
```

Final Arduino Uno Code

```
int LED = 2; //LED at Pin 2
int key = 0; //Default Key Value

void setup() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  Serial.begin(9600);

}

void loop() {
  if (Serial.available()) {
    key = Serial.read();
    Serial.write(key);
    morseCode();
  }
}

void morseCode() {
  if (key == 14) { //If enter key was pressed.
    digitalWrite(LED, HIGH);
  } else if (key == 15) { //If enter key was released.
    digitalWrite(LED, LOW);
  }
}
```

Bill of Materials

Part No.	Manufacturer	Description	Quantity	Direct Cost	Price / Package	Package Size (Units)	Notes
1056	Adafruit Industries	Laser	1	$18.95	$18.95	1	
PDB-C158	Advanced Photonix	Photodiode	1	$3.50	$3.50	1	
27979	Parallax	Parallax LCD	2	$85.98	$42.99	1	
A000066	Arduino	Arduino Uno R3	1	$17.94	$17.94	1	(on sale as of 11/17/2025)
A000062	Arduino	Arduino Due	1	$32.11	$32.11	1	(on sale as of 11/17/2025)
N/A	JLCPCB	2x3 PCB Proto. Boards	3	$1.80	$0.60	1	"package" is individual unit, not actual package
PN2222	onsemi	NPN Transistor	3	$0.90	$0.30	1	"package" is individual unit, not actual package
X001NHZMMB	Neosmuk	Magnetic Feet and Hooks	22	$15.38	$6.99	10	
N1P0810	Wiegmann	Metal Plate (8.25" * 6.25")	2.1	$17.85	$8.50	1	0.1 plate is used for the scope mount
FOS.2.2102R	SapiSelco	White Zipties	37	$0.46	$1.25	100	some longer zipties were used to secure the Arduino uno, but this is an equivalent
92000A220	(Unlisted)	M-4 Metric Screw for Magnetic Feet	12	$1.28	$10.69	100	(McMaster Carr Website)
98017A615	(Unlisted)	No. 6 Metal Washers	20	$1.00	$24.90	500	(McMaster Carr Website)
91755A141	(Unlisted)	Plastic Retaining Washers (For Securing Photodiode)	4	$0.70	$17.52	100	(McMaster Carr Website)
91772A155	(Unlisted)	6-32, 1.25" Screws for Board Mounting and Heat Sink	22	$2.66	$12.08	100	(McMaster Carr Website)
91841A007	(Unlisted)	6-32 Nut	24	$0.97	$4.04	100	(McMaster Carr Website)
94639A305	(Unlisted)	#6 Spacers	20	$1.98	$9.91	100	(McMaster Carr Website)
91792A151	(Unlisted)	6-32, 0.75" Screws for Photodiode Mount	2	$0.14	$7.19	100	(McMaster Carr Website)
B07B9X53N5	Higoo	Scope Mount for Laser	1	$35.95	$35.95	1	No listed manufacturer number. ASIN used instead (Amazon.com)
N/A	(Robert)	3D-Printed Adapter for Scope Mount	1	$0.20	$0.20	1	Printed by Instructor
MC7808CTG	onsemi	8V Voltage Regulator	2	$0.71	$17.79	50	(Used part number MC7808CT, but similar shown)
L7805CV	STMicroelectronics	5V Voltage Regalator	1	$0.29	$14.44	50	
11451	SparkFun Electronics	LED	3	$2.97	$0.99	1	
CF1/4C512J	KOA Speer Electronics	5k1 Resistor	3	$0.10	$33.95	1000	
CFS1/4C103J	KOA Speer Electronics	10k Resistor	3	$0.09	$57.50	2000	
270-AB	Wakefield Thermal Solutions	Heat Sink	2	$0.57	$285.04	1000	
E101MD1AV2BE	C&K	Toggle Switch	2	$17.57	$702.76	80	(can be ordered individually, but more expensive)
DN-R35S1-2	DINnector	DIN Rail	0.127	$0.57	$9.00	2	5" of rail (units in meters of rail)
179324-UK	Manhattan	Keyboard	1	$9.99	$9.99	1	
BC62766	Posdou	Adapter	1	$2.50	$4.99	2	
			Total	$275.11			

- Total cost for the project of about $275.11.
- **Note:** Parts with an "unlisted" manufacturer were found on McMaster Carr's website. No manufacturer was listed for these products, but no specific brand is needed, as they are generic parts (e.g. screws, washers, nuts, spacers, zipties).

Project Journal

Documentation Overview

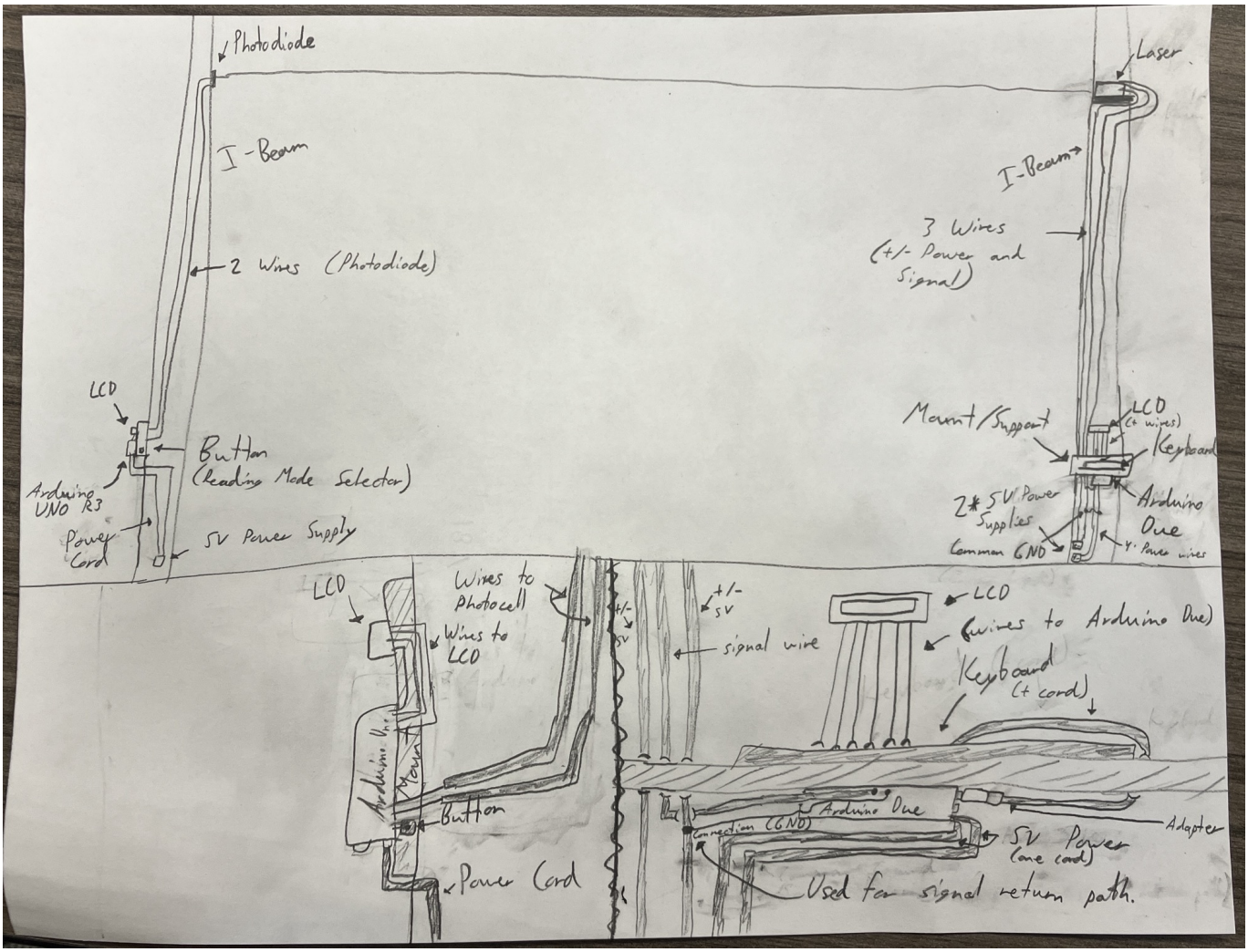
- Details for experiments listed in the "Experimentation" section.

09/26/2025

Completed Tasks

- Selected Project: Laser Communication System
- Explained Grading and Deadline Dates
- Identified Guidlines and Goals
  - Identified Conformance Criteria
  - Identified Functional Criteria
  - Identified Deliverables for Each All-Lab Day
  - Identified Stretch Goals
- Ordered Components

Rough Draft of Design



Deliverables and Guidlines

```
Wyatt Gleason,CSE=,102,TRM=,20250812,PRJ=,03,SCO=,??,DAT=,YYYYMMDD,COMMENT=,Project selected -- laser light comm system                                    # Due YYYYMMDD (All-Lab Day 07) -- grading/deadline dates, Construction Standards, conformance/functional criteria, prototype design, define deliverables, order parts
# Due YYYYMMDD (All-Lab Day 08) -- experiment with laser and light sensor over distances ; journal entry
# Due YYYYMMDD (All-Lab Day 09) -- experiment with USB keyboard to Arduino MCU interface and programming ; journal entry
# Due YYYYMMDD (All-Lab Day 10) -- experiment with Arduino outputting serial data to remote display via cable ; journal entry
# Due YYYYMMDD (All-Lab Day 11) -- experiment with Arduino outputting serial data to remote display via light ; journal entry
# Due YYYYMMDD (All-Lab Day 12) -- work day installing hardware in hallway ; journal entry
# Due YYYYMMDD (All-Lab Day 13) -- work day testing hardware in hallway ; journal entry
# Due YYYYMMDD (All-Lab Day 14) -- construction completed ; journal entry
# Due YYYYMMDD (All-Lab Day 14) -- Construction Standards inspection passed (-??% for retries)
# Due YYYYMMDD (All-Lab Day 14) -- Conformance tests passed (-??% for retries)
# Due YYYYMMDD (All-Lab Day 14) -- Functional tests passed (-??% for retries)
# Due YYYYMMDD (All-Lab Day 15) -- Submit Project Journal (includes schematic, code, test data, daily work entries, etc.)
# Due YYYYMMDD (All-Lab Day 15) -- Test-point/code challenge (-??% for retries)
# Due YYYYMMDD (All-Lab Day 15) -- additional = e-docs (+?%), abstract (+?%), BOM (+?%), UserGuide (+?%), impeccable (+?%)
# Functional test criteria
# --> Typing on keyboard results in text characters on the remote display
# --> STRETCH GOAL = Morse code option with sending key and remote lamp/LCD indicator
# Conformance test criteria
# --> Laser light cannot hit bystanders' eyes
# --> Keyboard and display at ergonomic heights and angles
# --> Must operate on 12-15 Volt DC power available in the hallway
# --> All typical wiring and soldering standards apply (see Construction Standards in course document)
```

09/29/2025

Completed Tasks

- Identified Relevant Construction Standards
- Completed Deliverables for All-Lab Day 07

Update to Deliverables

- All-Lab Day 08
    - Research and Present on Laser Safety

10/02/2025

Completed Tasks

- Researched and Presented on Laser Safety
- Made experimental plan for Laser and Photodiode experimentation.

10/06/2025

Completed Tasks

- Ran Laser and Light Over Distances Experiment and recoreded results.

10/07/2025

Completed Tasks

- Analyzed results for Laser and Light Over Distances Experiment.
    - Documentation of this experiment is listed in the "Experiment 28 - Choose Your Own" section.
- Submitted Project Journal Entry for All-Lab Day 08.
- Completed Deliverables for All-Lab Day 08.

10/16/2025

Completed Tasks

- Experimented with programs getting the Arduino Uno to display to an LCD, time based functions, and receiving input from an external module (Arduino Nano). Documented results and code are in the "Arduino Code Trials" section.

10/17/2025

Completed Tasks

- Experimented with communciating to Arduino Uno via cable. Used serial data to display text to screen. Documented results and code are in the "Arduino Code Trials" section.
- Submitted Project Journal Entry for All-Lab Day 09.
- Completed Deliverables for All-Lab Day 09.
- Deliverables re-arranged to put-off Arduino USB testing until Arduino Due arrives.

Update to Deliverables

- New Deliverables:

```
Wyatt Gleason,CSE=,102,TRM=,20250812,PRJ=,03,SCO=,??,DAT=,YYYYMMDD,COMMENT=,Project selected -- laser light comm system
# DONE 20250929 (before All-Lab Day 07) -- grading/deadline dates, Construction Standards, conformance/functional criteria, prototype design, define deliverables, order parts
# DONE 20251007 (before All-Lab Day 08) -- research and present on laser safety ; experiment with laser and light sensor over distances using signal generator as pulse source ; journal entry
# Due YYYYMMDD (All-Lab Day 09) -- experiment with Arduino outputting serial data to remote display via cable ; journal entry
# Due YYYYMMDD (All-Lab Day 10) -- experiment with Arduino outputting serial data to remote display via light ; journal entry
# Due YYYYMMDD (All-Lab Day 11) -- experiment with USB keyboard to Arduino MCU interface and programming ; journal entry
# Due YYYYMMDD (All-Lab Day 12) -- work day installing hardware in hallway ; journal entry
# Due YYYYMMDD (All-Lab Day 13) -- work day testing hardware in hallway ; journal entry
# Due YYYYMMDD (All-Lab Day 14) -- construction completed ; journal entry
# Due YYYYMMDD (All-Lab Day 14) -- Construction Standards inspection passed (-??% for retries)
# Due YYYYMMDD (All-Lab Day 14) -- Conformance tests passed (-??% for retries)
# Due YYYYMMDD (All-Lab Day 14) -- Functional tests passed (-??% for retries)
# Due YYYYMMDD (All-Lab Day 15) -- Submit Project Journal (includes schematic, code, test data, daily work entries, etc.)
# Due YYYYMMDD (All-Lab Day 15) -- Test-point/code challenge (-??% for retries)
# Due YYYYMMDD (All-Lab Day 15) -- additional = e-docs (+?%), abstract (+?%), BOM (+?%), UserGuide (+?%), impeccable (+?%)
# Functional test criteria
# --> Typing on keyboard results in text characters on the remote display
# --> STRETCH GOAL = Morse code option with sending key and remote lamp/LCD indicator
# Conformance test criteria
# --> Laser light cannot hit bystanders' eyes
# --> Keyboard and display at ergonomic heights and angles
# --> Must operate on 12-15 Volt DC power available in the hallway
# --> All typical wiring and soldering standards apply (see Construction Standards in course document)
```

10/23/2025

Completed Tasks

- Experimented with communicating to Arduino using Light.
    - Documented in "Signal Testing & Laser Light Communication" section.
- Discovered a functional technique for light communication.
- Broke a laser (unintentionally).

10/28/2025

Completed Tasks

- Experimented with communicating to Arduino with keyboard interface.
- Experimentd sending keystrokes from keyboard to other Arduino via cable.
- Documented in "Keyboard Input" section of Experiments.

10/29/2025

Completed Tasks

- Experimentd sending keystrokes from keyboard to other Arduino via laser light and photodiode.
- Documented in "Keyboard Input" section of Experiments.

10/30/2025

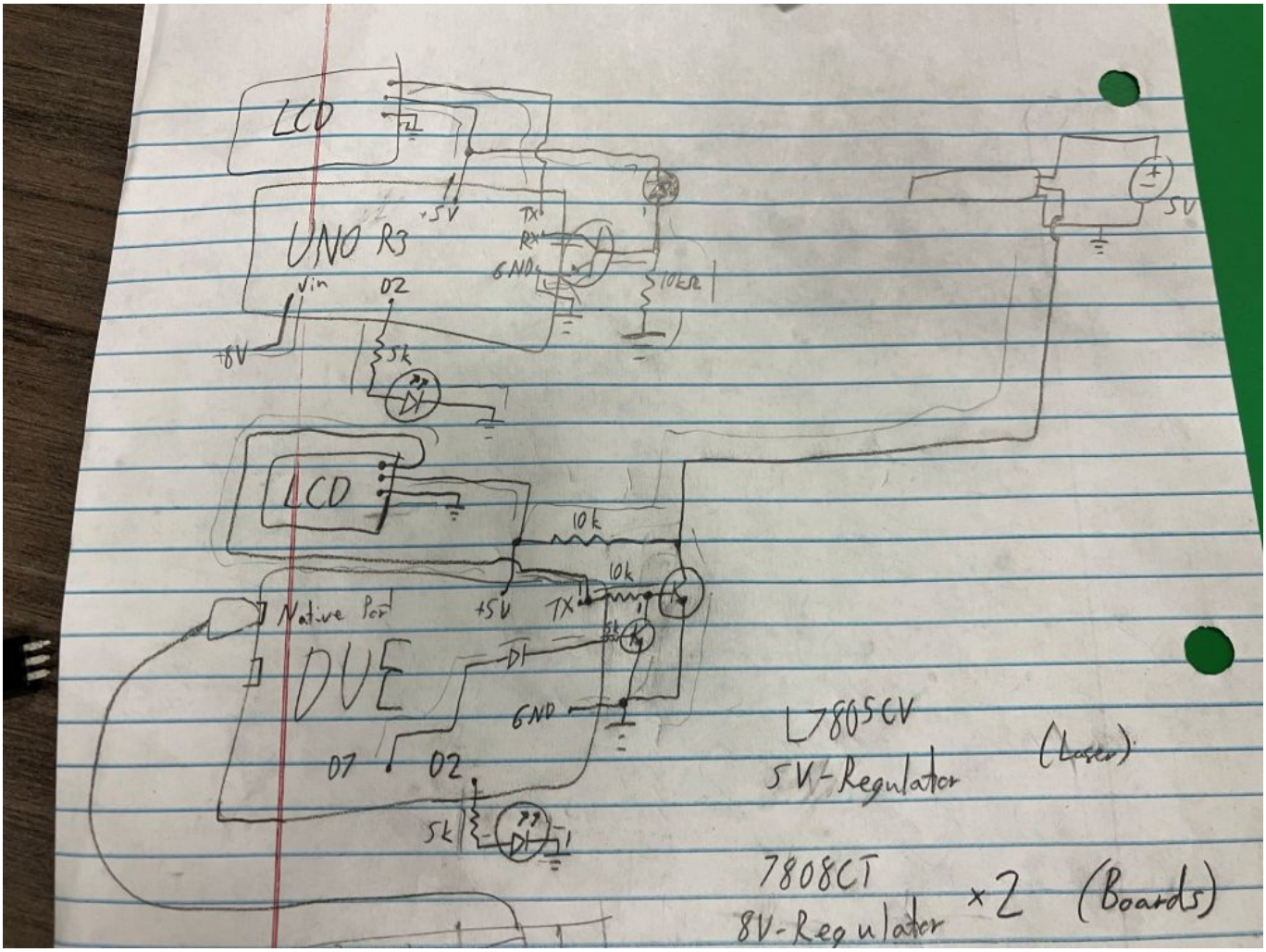
- Submitted project journal entry for All-Lab Day 11.
- Completed deliverables for All-Lab Day 11.

11/07/2025

Completed Tasks

- Worked on contructing circuit and equipment for hallway.
- Recorded hand-written drawing of schematic.
- Got and drilled into metal plates for permanent fixture.
- Submitted project journal for All-Lab Day 12.
- Completed deliverables for All-Lab Day 12.

Handwritten Schematic



- This is a hand-written schematic for my final construction.

11/10/2025

- Worked on construction
- Soldered wires to LCD 5V, GND, and RX
- Tested wire connection to LCD, and it didn't work.

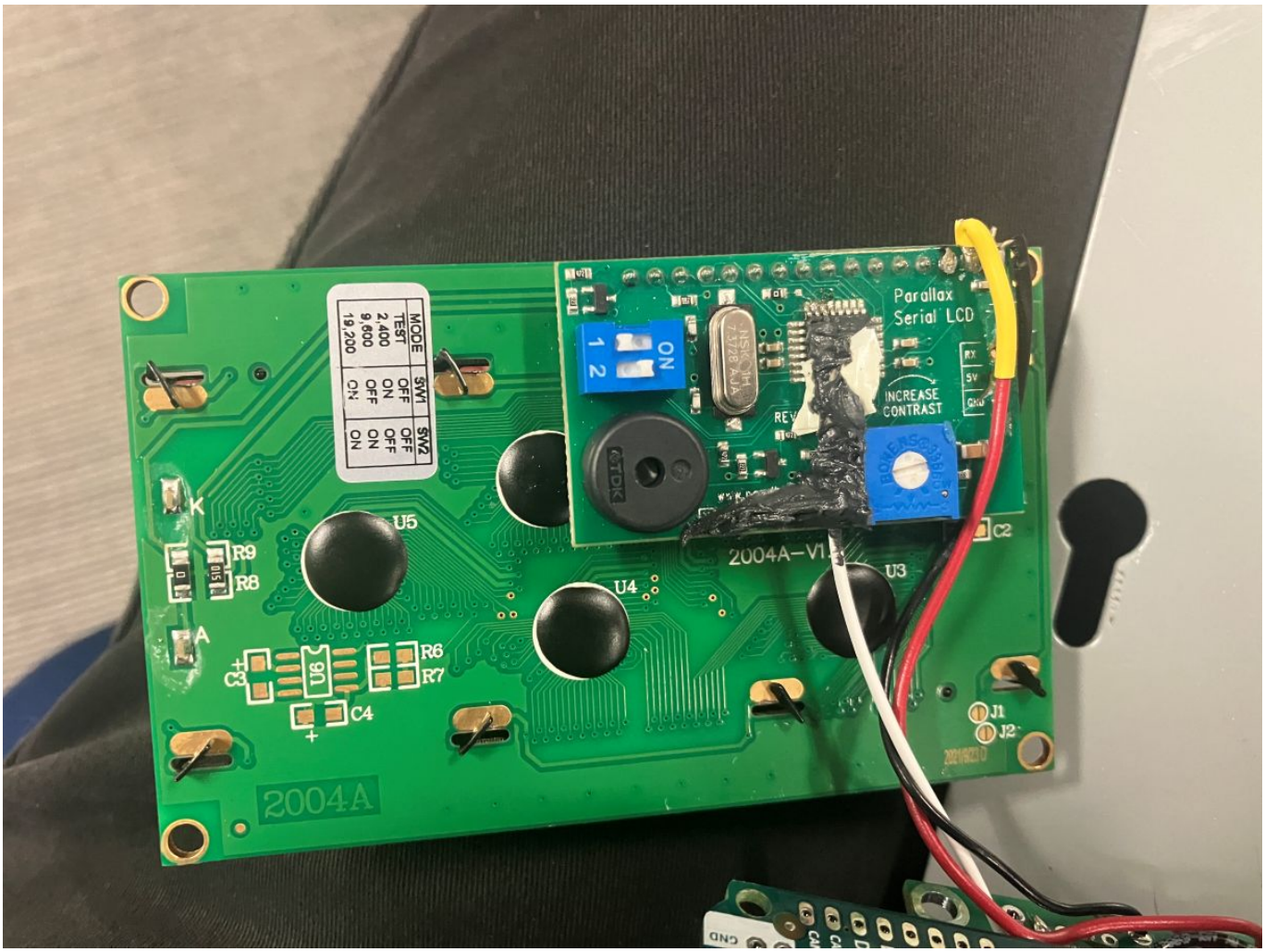
11/11/2025

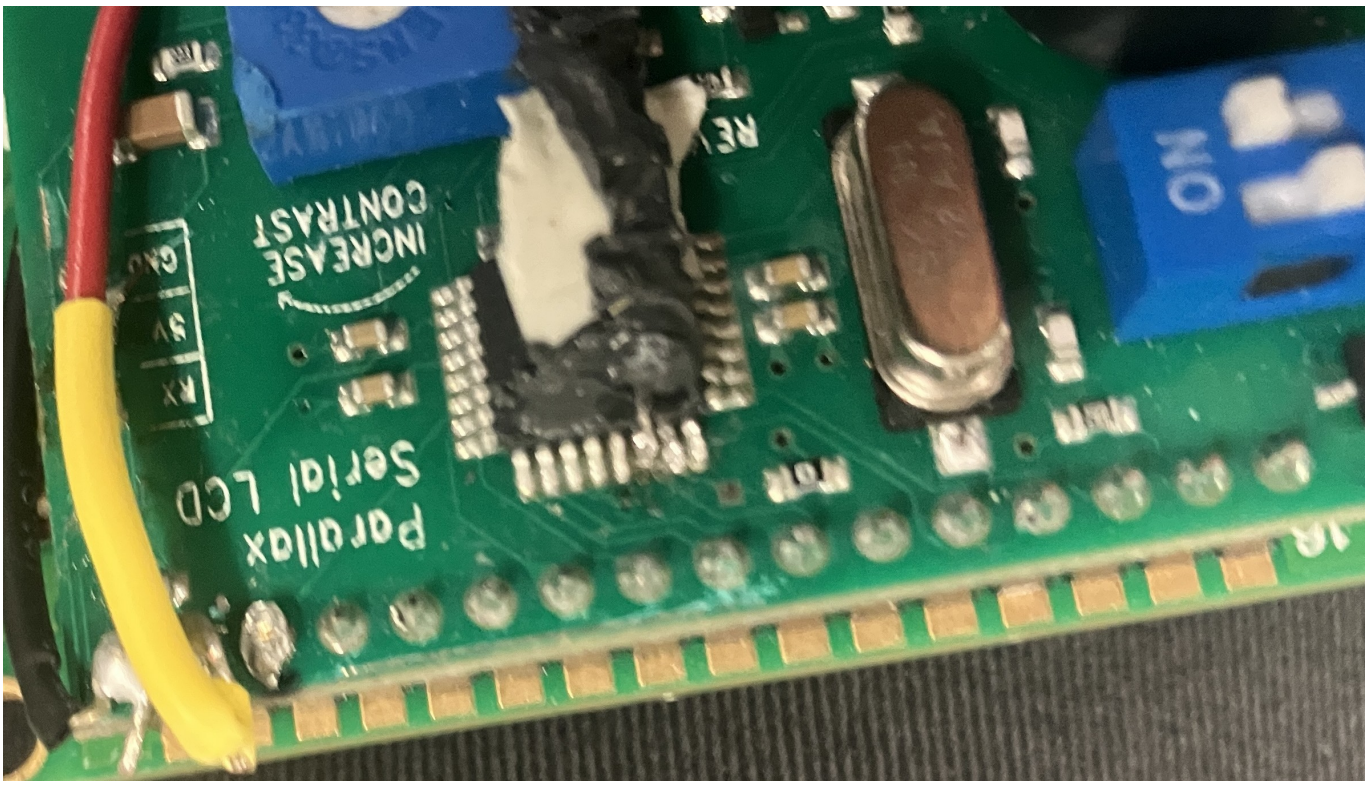
- Tried resoldering wires to LCD, and it failed. It resulted in the conductive pads on the board being destroyed.
- Worked around connection issues by attaching wires to electrically common points. Electrically common points were measured using an ohm-meter connecected with one probe on an exposed trace (solder-mask scraped away) and the other probe on the point being tested.
- 5V wire connected to LCD VDD temrinal. GND wire connected to VSS terminal. RX wire connected to third terminal (from the left side of the chip, shown below in "11/12/2025" journal entry) of a microchip on the driver circuit board.
- Applied J-B Weld to the wire to keep it from coming disconnected, as it did and I had to reconnect it.

11/12/2025

- J-B Weld finished curing.
- Tested LCD monitor, and it worked.
- Worked on further construction for hallway mounting.

Pictures





- J-B Weld is the dark gray paste along the board and white wire.
- Tape was used to temporarily secure the wire while the wire while the J-B Weld cured. This taped was thenn stuck in the J-B Weld afterwards (obviously), and so not all of the tape was remove.

- Also, this side of the LCD is not shown to the user of the project, and I didn't want to damage anything else, so I didn't try making the tape cut flush with the J-B Weld.

11/13/2025

- Worked on finalizing construction in hallway.
- Completed mounts and wiring for hallway setup.
- Tested laser-light system.

11/14/2025

- Finished construction of laser-light system.
- Submitted projet journal entry for All-Lab Day 13.
- Completed deliverables for All-Lab Day 13.
- Passed conformance test, functional test, and construction standards.
- Completed deliverables for All-Lab Day 14.

Experimentation

Documentation Overview

In this Experimentation Section:

Experiment 28 - Choose Your Own

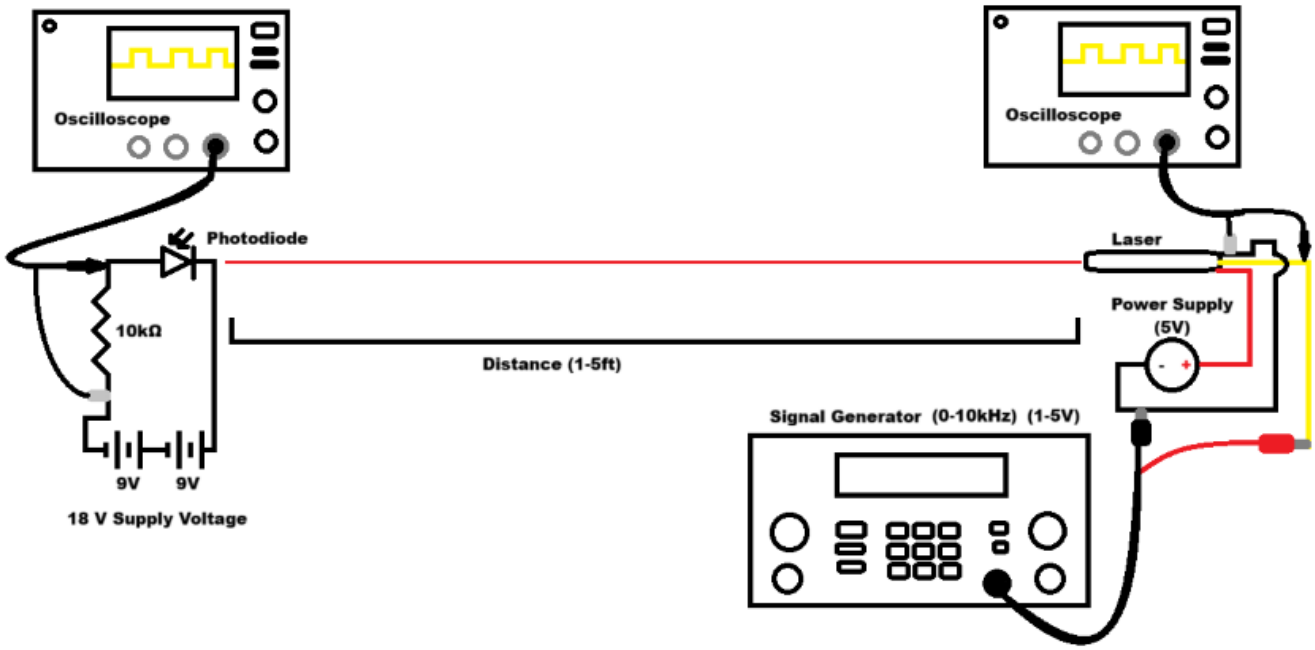
Experimentation

Laser & Light Over Distances

Stage 1

Hypothesis - I predict the laser will be reliable and stable at all test frequencies and voltage. The difference in voltage across a resistance due to distance will be negligible between a distance of 1ft and 5ft. Input voltage for stable laser output will be 3V or less.

Method



- Three factors will be changed:
  - Distance betweent the laser and the photodiode. (1-5ft, in 1ft increments.)
  - Frequency of laser on/off pulses. (0-10kHz: 10, 20, 50, 100, 200, 500, 1k, 2k, 5k, and 10k.)
  - Control voltage for the laser. (1-5V, in 1V increments.)
- Variance in distance.
  - The measured value will be the difference in voltage from 1ft versus the current distance, as measured from a peak to a valley on the sensor oscilloscope.
  - The laser frequency will be left at 1kHz.
  - The laser input voltage will be left at 5V.
- Variance in frequency.
  - The measured value will be how stable the signal is on the photodiode (receiving end) and the frequency of the laser as measured from the laser (to look for a loading or other noticeable effect). Values for laser measured frequency will only be recorded if a noticeable and significant difference occurs.
  - The signal generator will maintain an output voltage (for the laser input) of 5V.
  - The distance will stay at 1ft between laser and sensor.
- Variance in input voltage.
  - The measured value will be how the signal is affected on the sensor end, as a way of checking the minimal amount of power necessary for a stable pulsing control.
  - The frequency of the signal generator will remain at 1kHZ.
  - The distance will remain at 1ft between laser and sensor.

Predictions

Variable Distance						
Value	Distance	Voltage Peak	Voltage Value	Height Difference	Difference from 1ft Value	% Deviation
	1					
	2					
	3					
	4					
	5					
Units	ft	V	V	V	V	%

Variable Frequency							
Value	Freq.	Rise Time	Fall Time	Pulse Width	Rise % Width	Fall % Width	Consistent Pulse
	0.01						
	0.02						
	0.05						
	0.1						
	0.2						
	0.5						
	1						
	2						
	5						
	10						
Units	kHz	µS	µS	mS	%	%	Y/N

Variable Input Voltage							
Value	Input	Rise Time	Fall Time	Pulse Width	Rise % Width	Fall % Width	Consistent Pulse
	1						
	2						
	3						
	4						
	5						
Units	V	µS	µS	mS	%	%	Y/N

- It is not possible to predict how a product will react because of manufacturing differences or exact effects.
- The table above shows the data that will be recorded.
- The laser and photodiode will be reliable and ideal if the laser input can be 3V or less, the frequency can be at 60hZ or higher, and % voltage deviation at 5ft is less than 5%.
- I predict the laser will be able to handle ideal values for operation of my Laser Message Transmission System.

Risk Analysis

- Laser Emission
    - Do not stare into the laser. Doing so could result in retinal damage.

    - Do not point the laser into people's eyes. This laser is to be used as an instrument, not a weapon.
    - Good idea to point laser where no-one is "down-range" of the path, such as using a backdrop like a wall, and pointing the laser away from people.
- The laser I'm using is rated between 4.8V and 5.2V.
    - Do not exceed this rating, as it could ruin and damage the laser.
    - A good value to maintain is 5V, as it is right in-between the ratings.
    - Any values less than 4.8V will result in the laser not turning on.
- Oscilloscope Shorting
    - The oscilloscope is connected to earth ground through its reference lead. Do not connect the reference lead to any side of a component powered off the same power source of the oscilloscope, such side not being equipotential with ground.
    - Improperly connecting the reference lead could result in a short, damaging components, the oscilloscope, and possibly your body.
- Photodoide Voltage Rating
    - The photodiode is rated for a maximum reverse voltage of 30V. Do not exceed this voltage.
    - 18V is plenty enough for measuring photodiode response, while still being below 30V. In this case, two 9V batteries are being used to supply power to the photodiode circuit.
- Resistor Power Rating
    - Rating of Resistor is 250 mW.
    - A 10kΩ resistor with a voltage of 18V will experience approximately 32.4 mW, well below the max rating.
        - $P = {V^2 \over R} = {18^2 \over 10k} = 32.4$ mW
    - The resistor will not be over-powered.

Stage 2:

Data

Variable Distance						
Value	**Distance**	**Voltage Peak**	**Voltage Valley**	**Height Difference**	**Difference from 1ft Value**	**% Deviation**
	1	10.2	0	10.2	0	0.00%
	2	8.6	0	8.6	-1.6	-15.69%
	3	10.2	0	10.2	0	0.00%
	4	9.8	0	9.8	-0.4	-3.92%
	5	9.6	0	9.6	-0.6	-5.88%
Units	ft	V	V	V	V	%
	5.68V at 1ft, then 10.2V at 1ft					
	fluctuated a lot					
	centered vs. not					
	interference by blocking objects					
	most deviation caused by inconsistent moving					

Variable Frequency							
Value	Freq.	Rise Time	Fall Time	Pulse Width	Rise % Width	Fall % Width	Consistent Pulse
	0.01	0	0	45	0.00%	0.00%	Y
	0.02	0	0	24	0.00%	0.00%	Y
	0.05	0	0	10	0.00%	0.00%	Y
	0.1	0	0	5	0.00%	0.00%	Y
	0.2	0	0	2.5	0.00%	0.00%	Y
	0.5	0	0	1	0.00%	0.00%	Y
	1	1	1	0.5	0.20%	0.20%	Y
	2	1	1	0.25	0.40%	0.40%	Y
	5	4	4	0.104	3.85%	3.85%	Y
	10	4	4	0.054	7.41%	7.41%	Y
Units	kHz	µS	µS	mS	%	%	Y/N

Variable Input Voltage							
Value	Input	Rise Time	Fall Time	Pulse Width	Rise % Width	Fall % Width	Consistent Pulse
	1	4	4	0.5	0.80%	0.80%	Y
	2	4	4	0.5	0.80%	0.80%	Y
	3	4	4	0.5	0.80%	0.80%	Y
	4	4	4	0.5	0.80%	0.80%	Y
	5	4	4	0.5	0.80%	0.80%	Y
Units	V	µS	µS	mS	%	%	Y/N
	dramatic decrease in voltage below 1V input						

Analysis

- All math is calculated within the spreadsheet.
- The hypothesis was proven correct with variable frequency and input voltage, well within the prefered margins, and even exceeding expectations. The laser only needs just >1V to operate its driver, and can go up to 10kHz with minimal problems. The issue, however, is with variable distance. The result were too inconsistent to make an accurate analysis, and father distance testing would be beneficial, but the impact seemed to be negligble, as the effect would still be quite noticable at distance.
- Also, the noise from the lights did not effect the photodiode circuit as much as the laser, which is cool. (Tens of milliVolts for noise, and several whole Volts for laser effect.)

Variable Distance

- This test was by far the most challening, and I had inconsistent results.

- It was difficult to get the laser pointed at the exact same place on the photodiode as I moved it, and the location of the laser's dot on the photodiode greatly affected the output voltage.
- When I first tested at 1ft, I got 5.68V out. After a second test (realigning the laser with the photodiode) I got 10.2V out.
- The variation (or % deviation) is mostly due to inconsistent testing, and not distance.
- The exact effect on voltage output is unclear, but it was evident from watching the output voltage versus photodiode noise that the difference is noticable, and could be picked up by a microcontroller on the other end.
- I would need a vary stable test method and setup to confirm the actual effects of distance, but it appears from other tests the the voltage output is significant enough for my project.

Variable Frequency

- Very consistent and accurate results.
- I originally had the output circuit incorrectly made. As to how, I don't know, but it produced weird result. The circuit is shown in the photos section, along with the results.
- I also originally had the wrong input frequency (imprecise quantities by about +/- 5%) and voltage (not 5V, but around 2-3V). However, after redoing the test I found that the variance (if any) on the output circuit was negligible if even noticable at all (especially for input voltage).
- Interestingly, as the frequency got lower, the pulse width of the "on" time was not directly proportional with the decrease in frequency (by a factor of 10).
- The laser emitted a conssitent pulse over all frequencies, and is well suited for my project, as I only need a consistent pulse of about 40Hz to meet my goal.

Variable Voltage Input

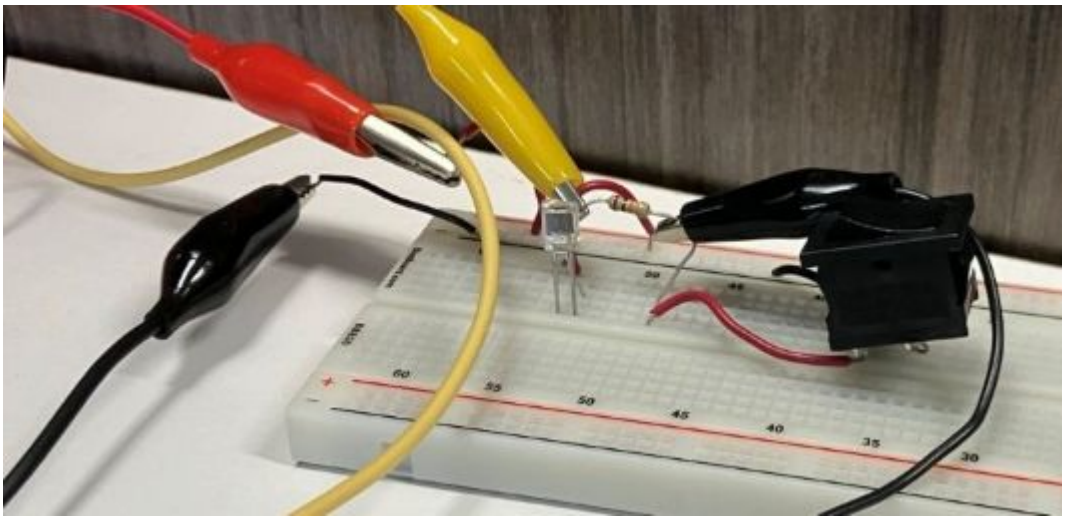
- The voltage input from 1-5V had negligible impact on the laser, as measured from the output circuit.
- The photodiode ouptut circuit produced almost the same exact voltage output across all laser driver input voltages, and any difference was barely noticeable.
- The laser did, however, fail as the voltage input dropped below 1V, with a slight decrease in input voltage dramatically affecting the laser's output, and thus the voltage output on the receiving end. (Shown in photos section below.)
- Minimum input voltage is 1V, but that is risking possible failure.
- I will keep a voltage input of 2V, which is significantly above minimum operating while still staying under the 3.3V max rating or the Arduino Due microcontroller, used for laser input.

Issues

- Holding the laser steady on the photodiode while changing its position was extremely difficult.
- I had an incorrect circuit setup for the original test, before I rearranged the circuit and it worked correctly.

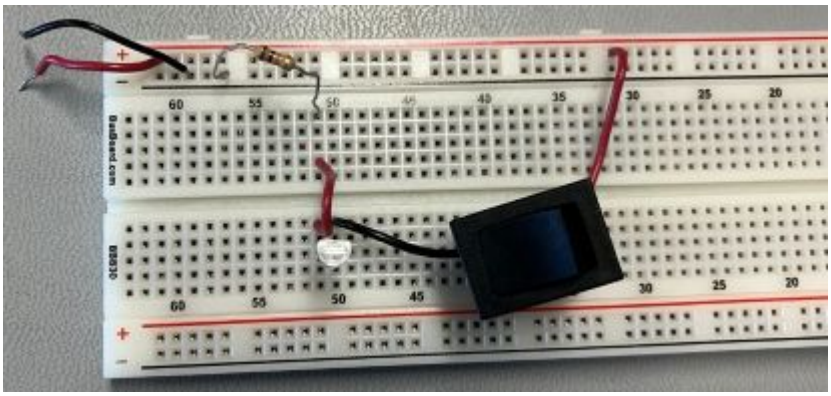
Photos

Origianl Circuit



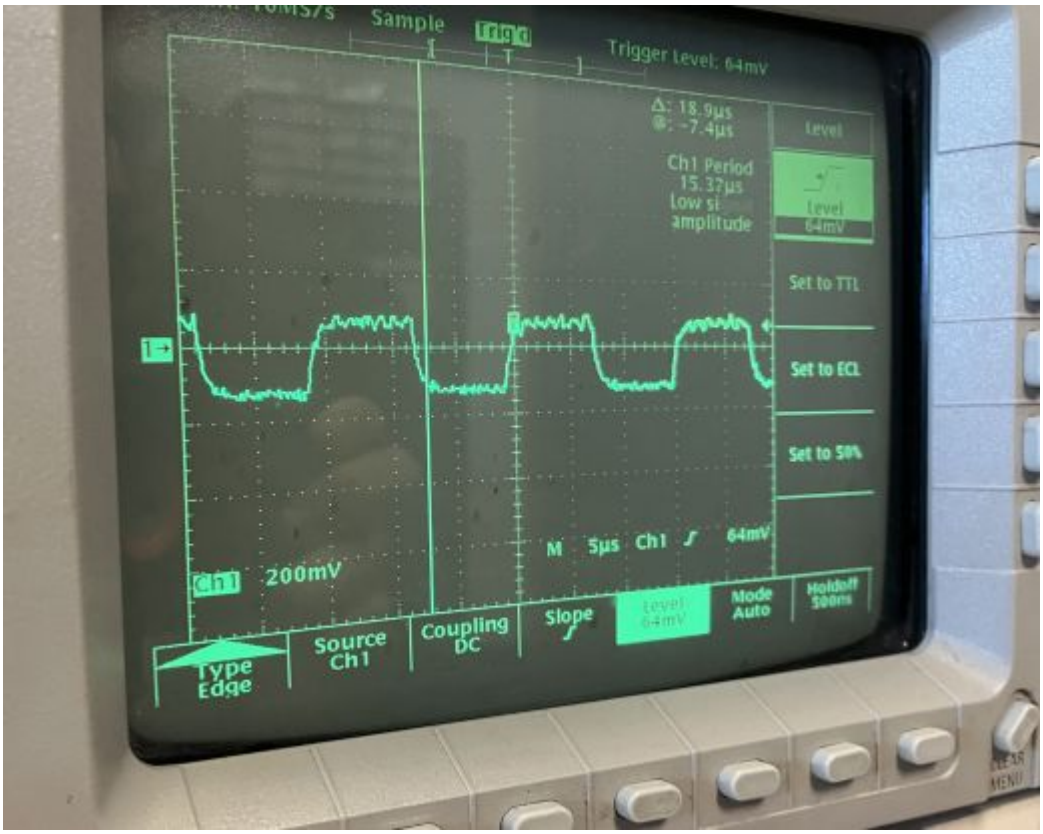
- The only difference in my original testing circuit versus this one is that the photodiode had to be flipped - pointing the opposite direction - ot work properly, which made it extremely difficult to shine the laser into. This difficulty is the main reason I reconstructed the circuit.

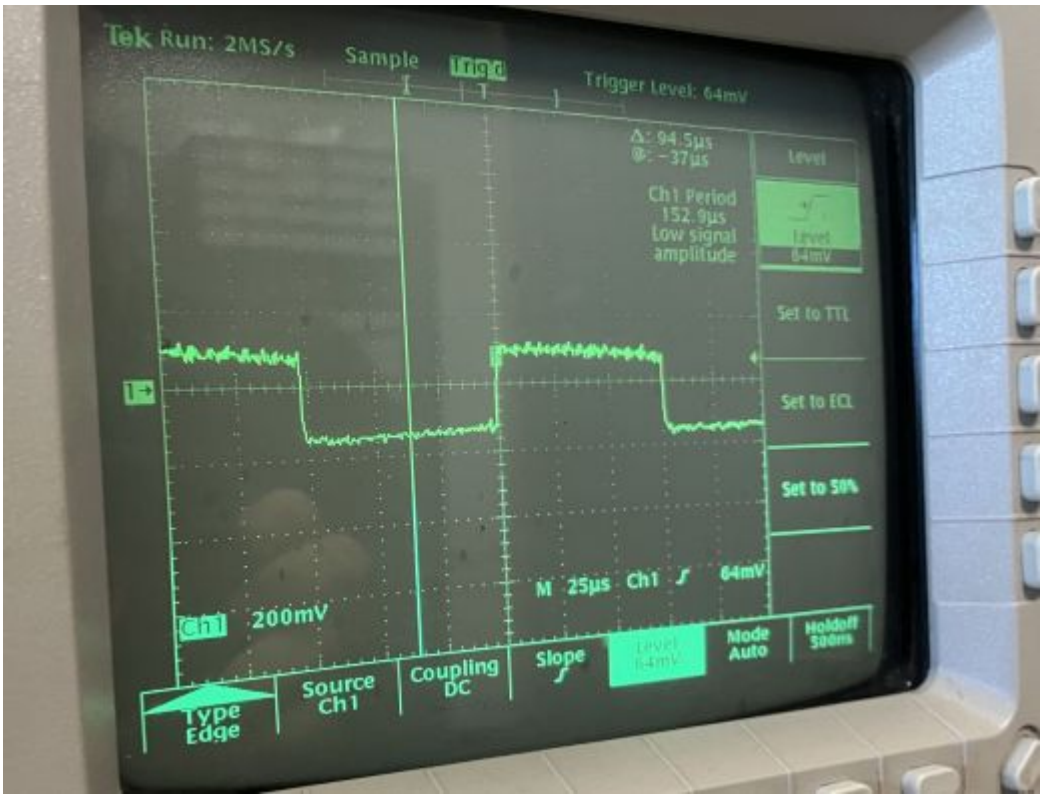
New Circuit (Test Circuit)

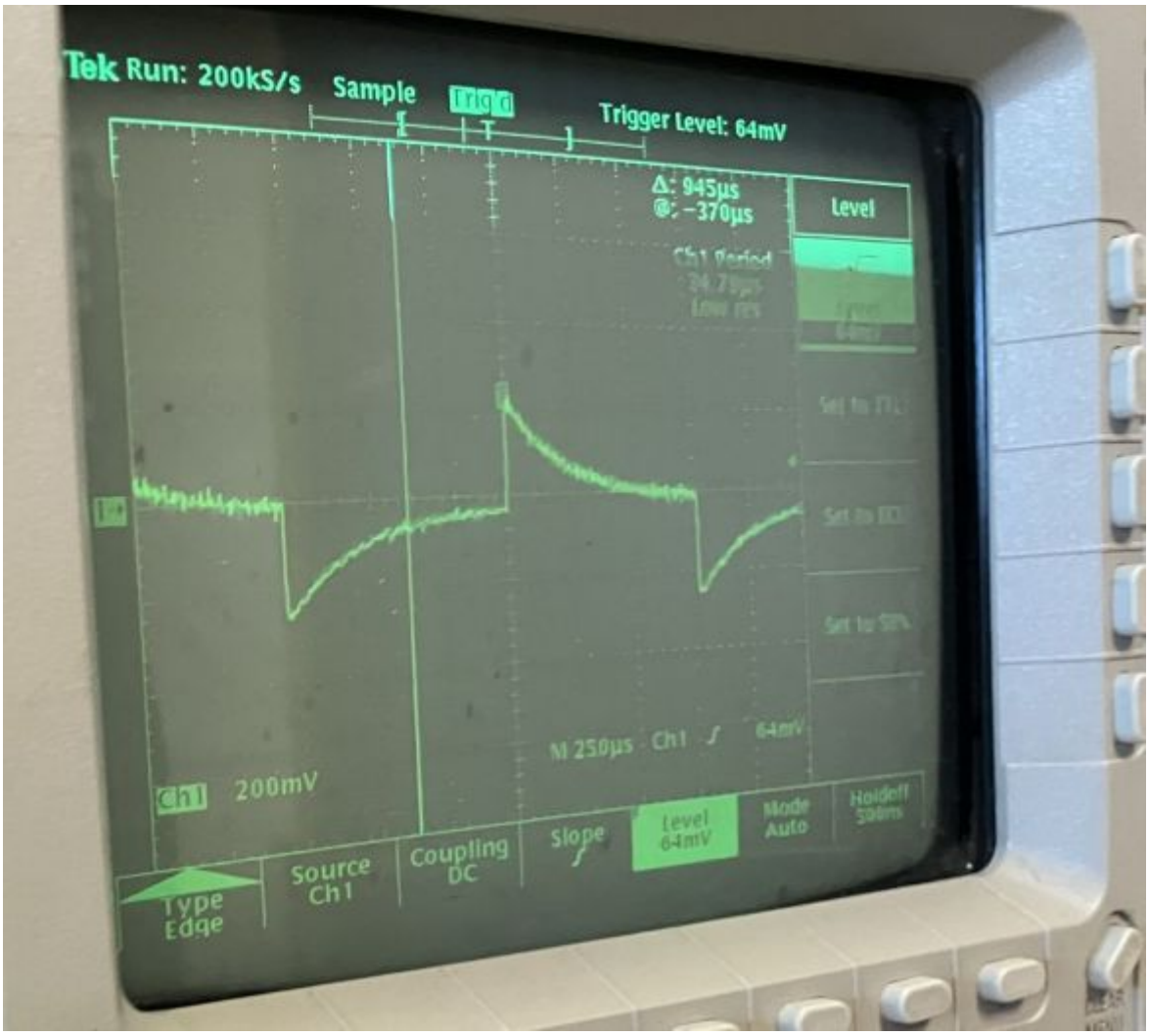


- This it the circuit I used during the recorded testing results. (Black to negative, red to positive.)

Original Circuit Unstable Values

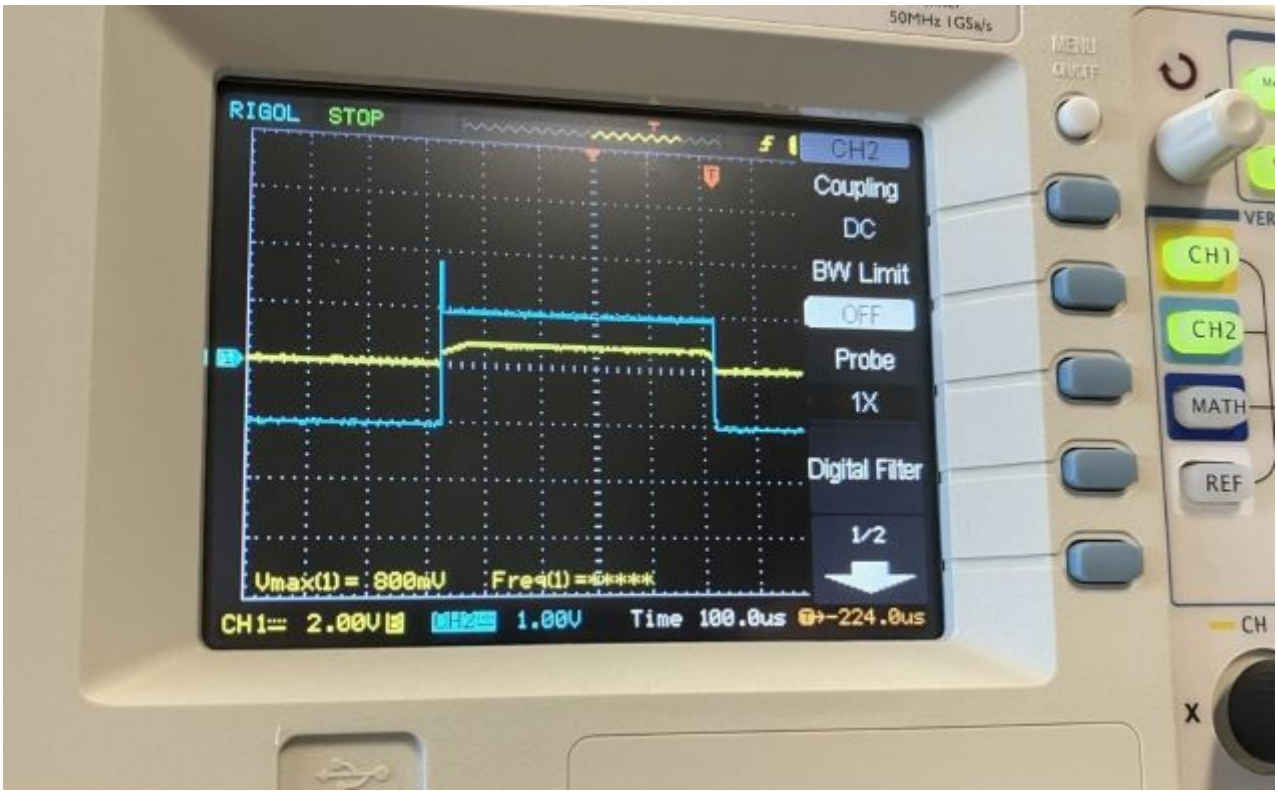

Sample
Trigd
Trigger Level: 64mv
Δ: 18.9µs
@: -7.4µs
Ch1 Period
15.37µs
Low si
amplitude
Level
Level
64mV
Set to TTL
Set to ECL
Set to 50%
M 5µs Ch1 ∫ 64mV
Ch1 200mV
Type
Edge
Source
Ch1
Coupling
DC
Slope
Level
64mV
Mode
Auto
Holdoff
500ns

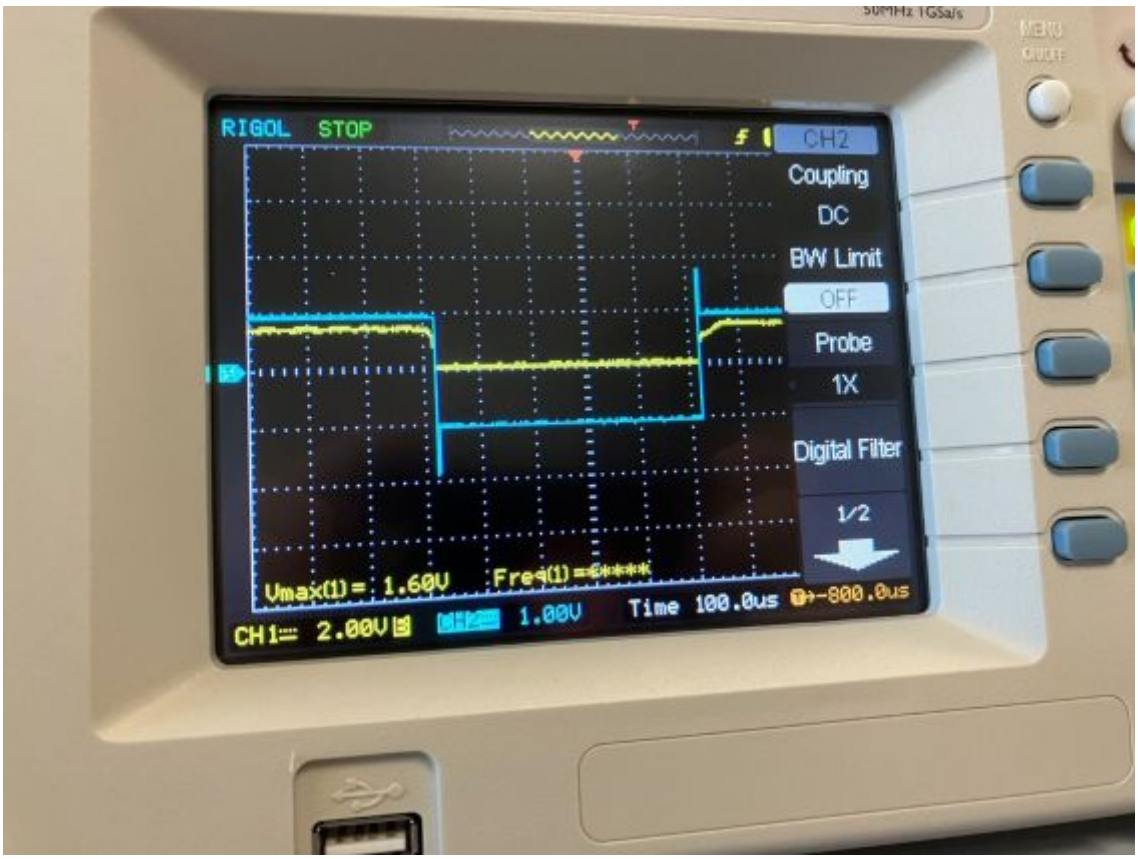

Tek Run: 2MS/s
Sample
Trigd
Trigger Level: 64mv
Δ: 94.5µs
@: -37µs
Ch1 Period
152.9µs
Low signal
amplitude
Level
Level
64mV
Set to TTL
Set to ECL
Set to 50%
M 25µs Ch1 ∫ 64mV
Ch1 200mV
Type
Edge
Source
Ch1
Coupling
DC
Slope
Level
64mV
Mode
Auto
Holdoff
500ns

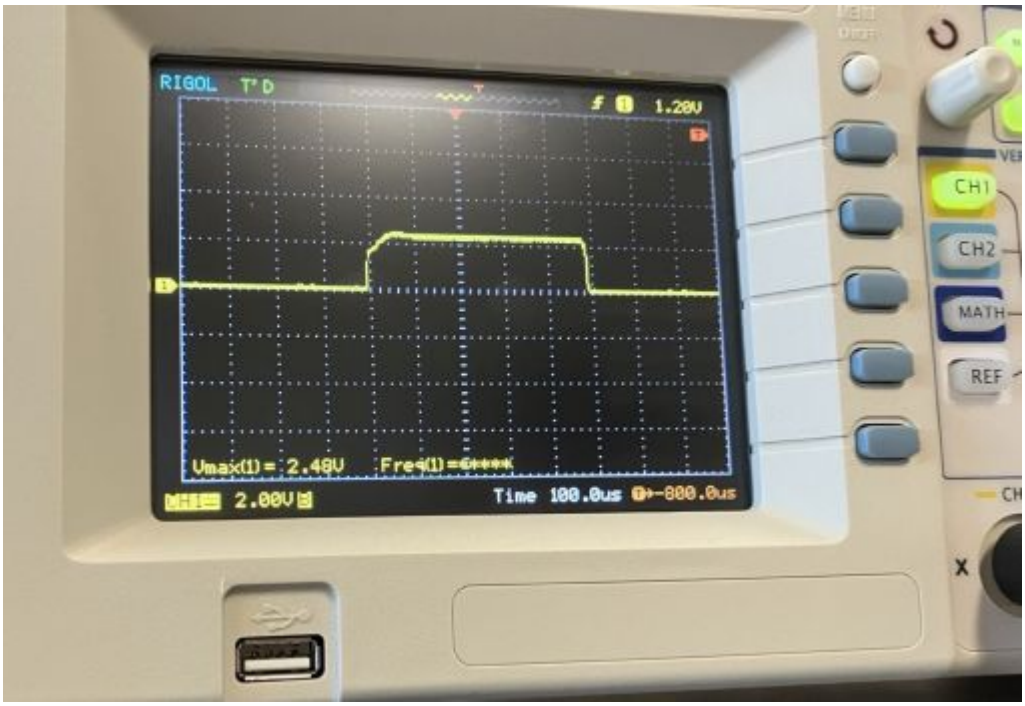


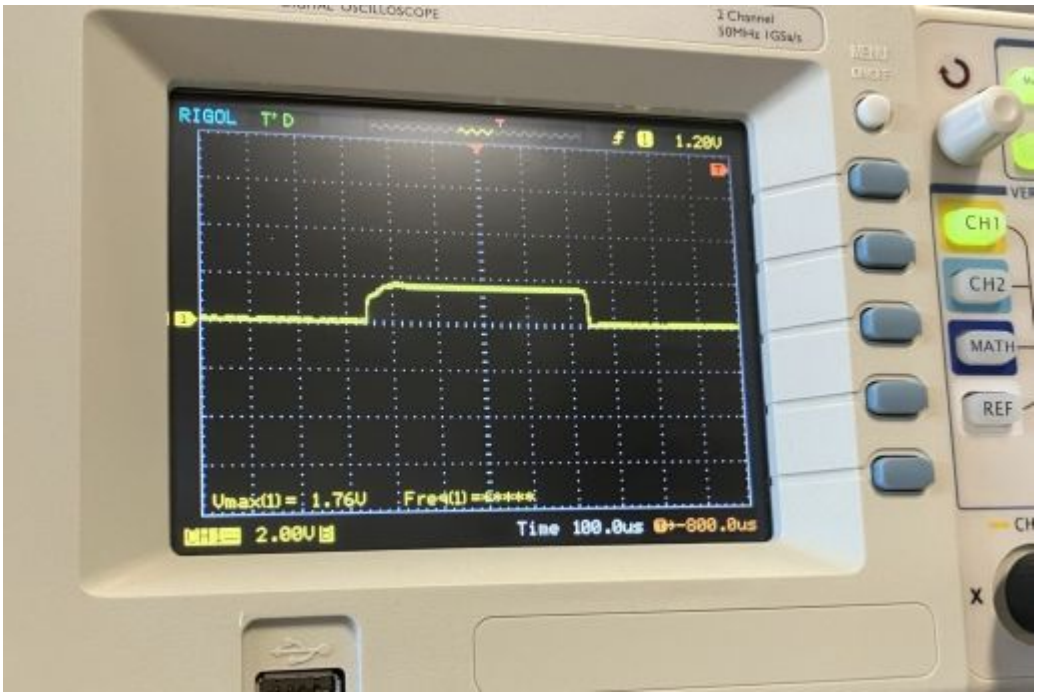
- Different frequency testing, but very unstable pulses. They were supposed to be square waves, but became artificial square waves, rather than actual. At a lower frequency it became a triangle wave (only stable at high frequencies).

Voltage Sagging


50MHz 1GSa/s
RIGOL STOP
CH2
Coupling
DC
BW Limit
OFF
Probe
1X
Digital Filter
1/2
Vmax(1)= 800mV Freq(1)=****
CH1 2.00V
1.00V
Time 100.0us -224.0us
MENU
CH1
CH2
MATH
REF
CH
X

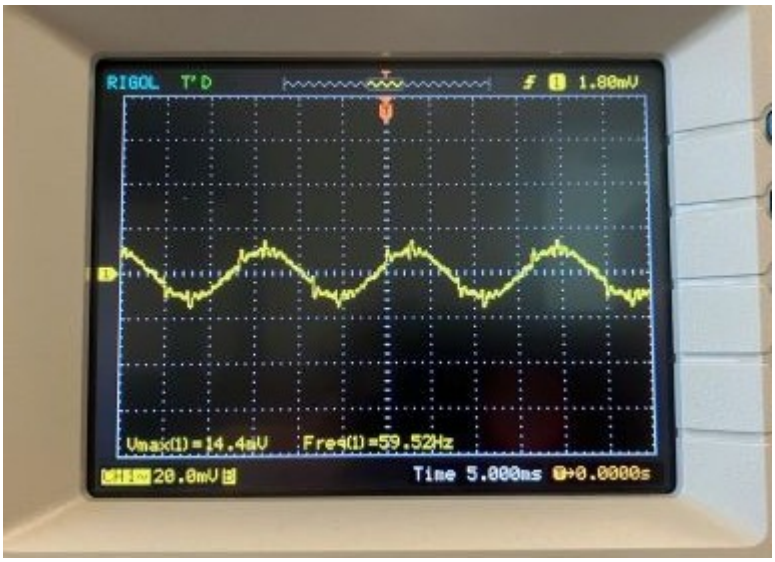

RIGOL STOP
CH2
Coupling
DC
BW Limit
OFF
Probe
1X
Digital Filter
1/2
Vmax(1)= 1.60V Freq(1)=****
CH1 2.00V
1.00V
Time 100.0us -800.0us





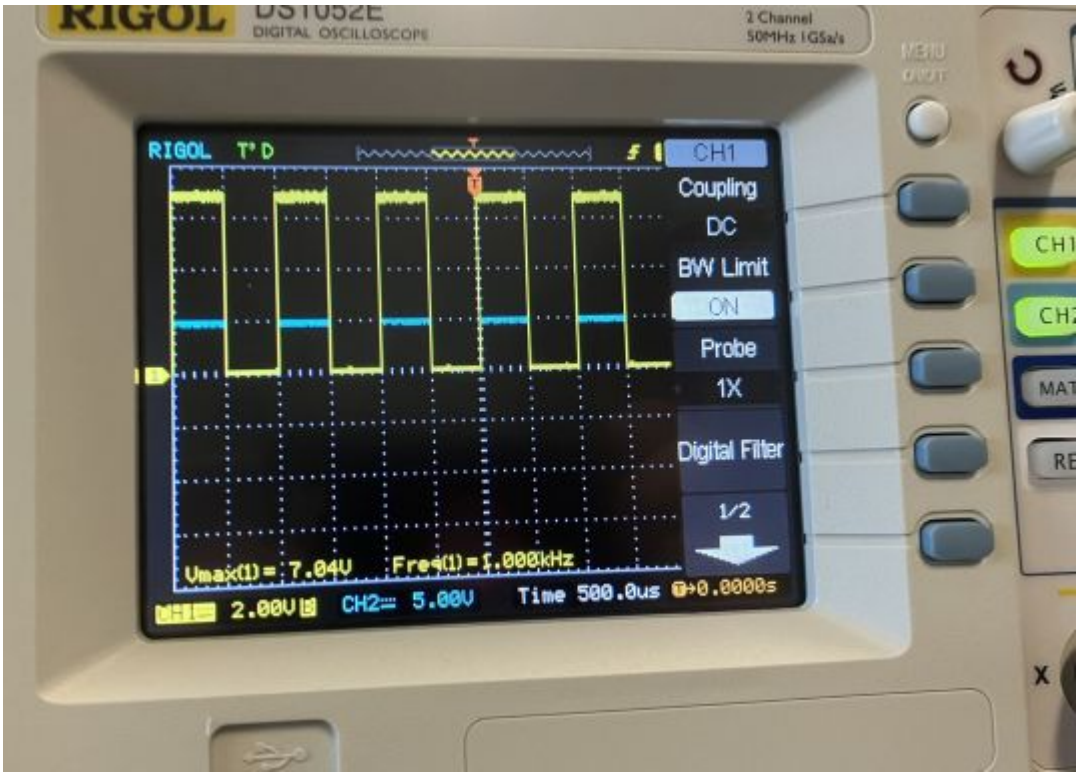
- Blue line represents voltage input to laser driver, yellow line represents output from photodiode circuit.
- When the input got below 1V, the voltage output sagged significantly.
- As you can see on the yellow line, the wave becomes rounded on its edges as the input to the laser driver decreases.

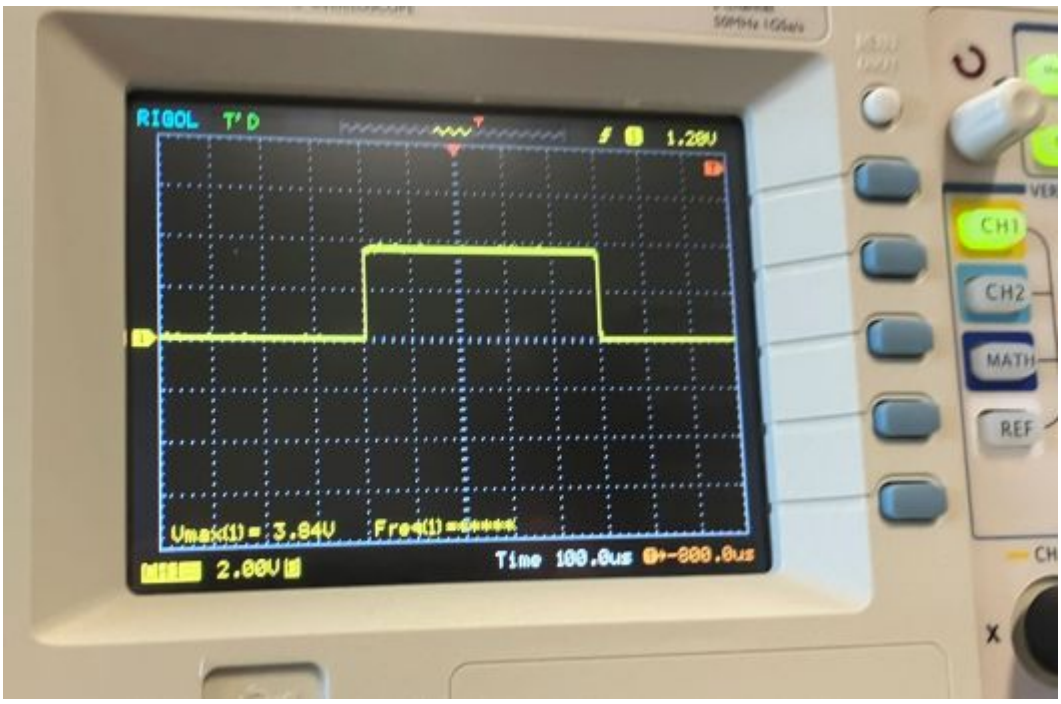
Background Noise

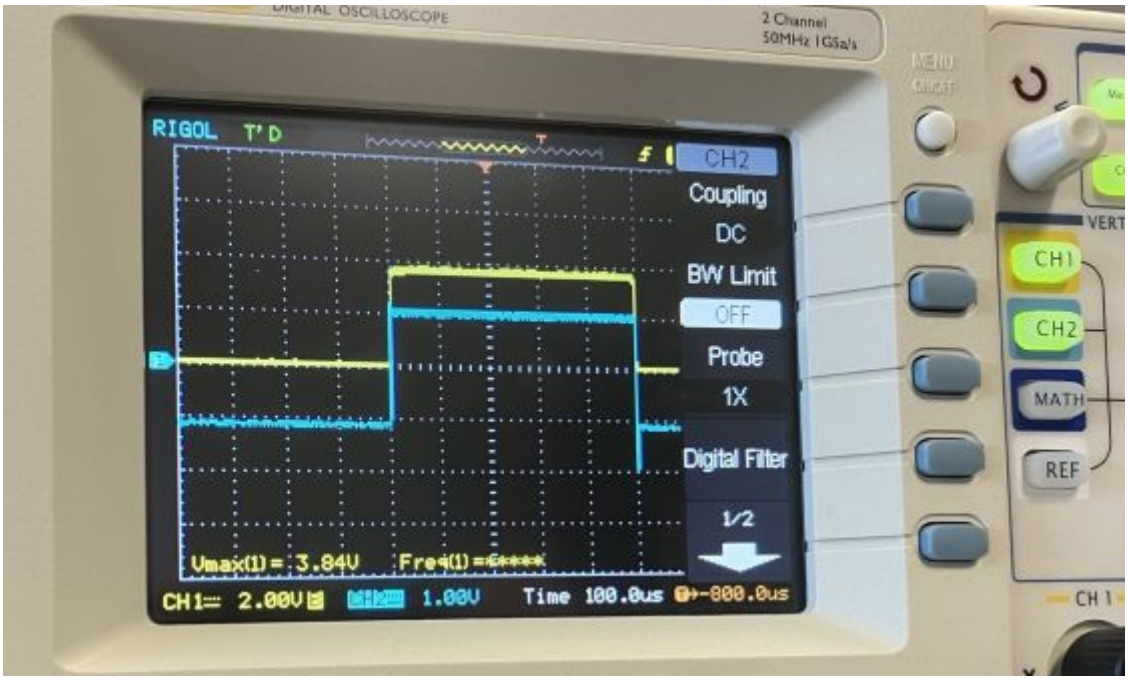


- Background noise of original circuit. It has a frequency of 60Hz, the same as the lights and electrical system in the building.

Stable Pulsing Example of Test Circuit


RIGOL
DS1052E
DIGITAL OSCILLOSCOPE
2 Channel
50MHz 1GSa/s
RIGOL T'D
CH1
Coupling
DC
BW Limit
ON
Probe
1X
Digital Filter
1/2
Vmax(1)= 7.04V
Freq(1)= 1.000kHz
2.00V
CH2 5.00V
Time 500.0us
0.0000s
CH1
CH2
MAT
RE
X


RIGOL T'D
1.20V
Vmax(1)= 3.84V
2.00V
Time 100.0us
CH1
CH2
MATH
REF
X



- Again, blue for input and yellow for output.
- Very stable pulses.
- These photos were taken while using a stable input on the newer circuit, the one used for the recorded tests.
- Interestingly, the voltage input seems to spike at the edges of the wave, but it is very brief.

Arduino Code Trials

Experimentation

Arduino LCD Output

First Test Run of LCD

```
#include <ParallaxLCD.h>

ParallaxLCD lcd(2,2,16); // desired pin, rows, cols

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up
  lcd.at(1,3,"Hello World");
}
```

- This didn't even compile. The arduino IDE wanted to have a loop function, even if it was empty.
- Adding "loop" function:

```

```

```
#include <ParallaxLCD.h>

ParallaxLCD lcd(2,2,16); // desired pin, rows, cols

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up
  lcd.at(1,3,"Hello World");
}

void loop () {
}
```

- Compiled, but didn't display properly. The most I saw was a moving cursor and a "(" in the bottom right corner.
- Changing rows to 4 and columns to 20 to match parallax LCD rows and columns.

```
#include <ParallaxLCD.h>

ParallaxLCD lcd(2,4,20); // desired pin, rows, cols

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up
  lcd.at(1,3,"Hello World");
}

void loop () {
}
```

- No characters, only a cursor that moved.

> Blank Cursor:
>
> 

- I even tried running a provided example from the Parallax LCD Examples in Arduino IDE:

```
#include <ParallaxLCD.h>

// Change these values if you're using the 4x20 LCD
#define ROWS 2
```

```
#define COLS 16

ParallaxLCD lcd(2,ROWS,COLS); // desired pin, rows, cols

/**
 * Initialize the 2-dimensional custom character byte array.
 * Even though each 'byte' is 0-255, only the lowest 5 bytes are used. So only
 *  0-31 are valid values.
 */
byte customCharacters[8][8] = {0, 4, 14, 31, 14, 4, 0, 0,   // Diamond
                               0, 10, 14, 31, 31, 14, 4, 0, // Heart
                               0, 4, 14, 31, 14, 4, 14, 0,  // Spade
                               4, 14, 4, 10, 31, 10, 4, 14, // Club (sorta...)
                               4, 14, 21, 4, 4, 4, 4, 4,    // Up
                               4, 4, 4, 4, 4, 21, 14, 4,    // Down
                               27, 22, 13, 27, 22, 13, 27, 22, // Stipple pattern
#1
                               29, 14, 23, 27, 29, 14, 23, 27  // Stipple pattern
#2
                             };

void setup () {

  lcd.setup();
  delay(1000);
  lcd.backLightOn();
  lcd.empty();

  // Count some milliseconds
  lcd.at(1,4,"Milliseconds\0");
  delay(1000);
  lcd.off();
  delay(1000);
  lcd.on();
  lcd.at(0,0,"m:\0");
  for (int b=0; b<=100; b+=5) {
    lcd.at(0,2,millis());
    delay(500);
  }

  // Line feeds
  lcd.empty();
  lcd.print("Line Feed\0");
  for (int b=0; b<=50; b+=5) {
    lcd.lf();
    delay(500);
  }

  // Carriage Returns
  lcd.empty();
  lcd.print("Chr Return\0");
  delay(1000);
  for (int b=0; b<=30; b+=5) {
    lcd.pos(0,10);
```

```
    delay(500);
    lcd.cr();
    delay(500);
  }

  // Different cursors
  lcd.empty();
  lcd.cursorBlock();
  lcd.print("Block Cursor\0");
  for (int x=0; x<COLS; x++) {
    lcd.pos(1,x);
    delay(500);
  }
  lcd.empty();
  lcd.cursorUnderline();
  lcd.print("Underline Cursor\0");
  for (int x=0; x<COLS; x++) {
    lcd.pos(1,x);
    delay(500);
  }
  lcd.backLightOff();
  lcd.cursorOff();

  // Clear the screen and enjoy a brief interlude.
  lcd.empty();
  lcd.print("Music Maestro!\0");
  lcd.playTone(213, 216, 223);
  lcd.playTone(213, 216, 223);
  lcd.playTone(214, 216, 227);
  lcd.playTone(210, 217, 220);
  lcd.playTone(210, 217, 222);
  lcd.playTone(210, 217, 224);
  lcd.playTone(210, 217, 226);
  delay(3000);

  /**
   * Initialize the custom character slots on the LCD with the predefined values.
   */
  int i;
  for(i=0; i < 8; i++) {
    lcd.setCustomCharacter(i, customCharacters[i]);
  }
  lcd.empty();
  lcd.backLightOn();
  // Now display them for 2 seconds each.
  for(i=0; i < 8; i++) {
    lcd.at((i%ROWS),0, "Custom Char \0");
    lcd.at((i%ROWS),13, i);
    lcd.printCustomCharacter(i);
    lcd.cr();
    delay(2000);
  }
}
```

```
void loop () {
  lcd.at(ROWS,9,millis());
}
```

- This turned out to be a garbled mess and nothing made sense. Also, the buzzer on the back was going off and that was annoying.

> Garbage Text:
>
> 
>

- It turns out I had a switch on the LCD module in the wrong position. After flipping S1 into the "on" position, the code worked properly, and I read "Hello World!". I first used "A" as a print value when troubleshooting it, but it displayed both values as requested. The combination of S1 and S2 controlls the "baud" rate of the LCD module, or the rate at which it is looking for and recieves information.

> Not Working:
>
> 
>
>
> Working:

"Hello World"

Note: S1 and S2 have to both be in the "ON" position for the LCD to work properly.

Updating Value and String

```
#include <ParallaxLCD.h>

ParallaxLCD lcd(2,4,20); // desired pin, rows, cols

int x = 1;
String text = String(x); //Convert x to String and assign it to "text"

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up
  lcd.empty(); // Clear the screen
  lcd.at(1,3,x); //Display the value of x
}

void loop () {
  delay(1000);
  x += 1;
  text += x; //Concatenate current string and value of x
  lcd.at(1,3,x);
  lcd.at(2,3, text); //Display the current string of characters
}
```

- Worked as intended.
- Screen was cleared prior to writing to it, value of x was displayed, and the total string was also displayed.
- It might be more efficient to parse the string and add the character froms there.

Parsing Characters

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
ParallaxLCD lcd(2, ROWS, COLS); // desired pin, rows, cols

String text = "This is a sentence. It is supposed to wrap after reaching the end
of the line. Hopefully this works.";
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {
  delay(100);
  if (text.length() > 0) { //Only update values if there is a value to update.
(Don't try to use a null string.)
    lcd.at(row, col, text.substring(0, 1)); //Display the first character of the
string.
    text = text.substring(1); //Shorten the string.
    col++; //Advance position.
    if (col > COLS - 1) { //Reset col if value exceeds max columns, and advance
row.
      col = 0;
      row++;
    }
    if (row > ROWS - 1) { //Reset row if value exceeds max rows, and reset screen.
      row = 0;
      lcd.empty();
    }
  }
}
```

- Worked as intended, wrapping characters around the display and reseting it when the row was exceeded.
- However, it would be betteer if words were not split, but remained together.

- Look for space then parse:

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
ParallaxLCD lcd(2, ROWS, COLS); // desired pin, rows, cols

String text = "This is a sentence. It is supposed to wrap after reaching the end
of the line. Hopefully this works.";
String curWord = "";
int spaceIndex = 0;
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  delay(100);
  if (text.length() > 0 || curWord.length() > 0) { //Only update values if there
is a value to update. (Don't try to use a null string.)

    if (curWord.length() < 1) { //Only update current word when empty.
      spaceIndex = text.indexOf(" ");
      if (spaceIndex > -1) { //Update current word up to spaceIndex if there is a
space, else, just make the current word be the remaining text.
        curWord = text.substring(0, spaceIndex);
        text = text.substring(spaceIndex + 1);
      } else {
        curWord = text;
        text = "";
      }
    }

    if (col + curWord.length() > COLS - 1 || col > COLS - 1) { //Look for either
to word being too long or the column position offscreen.
      col = 0;
      row++;
      if (row > ROWS - 1) { //Reset row if value exceeds max rows, and reset
screen.
        row = 0;
        lcd.empty();
      }
    } else {
      if (curWord.length() == 1) {
         lcd.at(row, col, curWord.substring(0, 1) + " "); //Display the character
of the string plus a space.
         col += 2;
```

```
      } else {
         lcd.at(row, col, curWord.substring(0, 1)); //Display the character of the
string.
         col++;
      }
      curWord = curWord.substring(1); //Shorten the string.
    }

  }

}
```

- Worked as intened after some trial and error.

Getting LCD to use Time instead of Delay

- I need the Arduino to be able to accpet other input while uploading the text at the same time. Using the delay function pauses all other actions for a specified time. I will have to use to get around this issue.
- Using time instead of delay:

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
ParallaxLCD lcd(2, ROWS, COLS); // desired pin, rows, cols

String text = "This is a sentence. It is supposed to wrap after reaching the end
of the line. Hopefully this works.";
String curWord = "";
int spaceIndex = 0;
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

void setup () {
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  delay(100);
  if (text.length() > 0 || curWord.length() > 0) { //Only update values if there
is a value to update. (Don't try to use a null string.)

    if (curWord.length() < 1) { //Only update current word when empty.
      spaceIndex = text.indexOf(" ");
      if (spaceIndex > -1) { //Update current word up to spaceIndex if there is a
space, else, just make the current word be the remaining text.
```

```
        curWord = text.substring(0, spaceIndex);
        text = text.substring(spaceIndex + 1);
      } else {
        curWord = text;
        text = "";
      }
    }

    if (col + curWord.length() > COLS - 1 || col > COLS - 1) { //Look for either
to word being too long or the column position offscreen.
      col = 0;
      row++;
      if (row > ROWS - 1) { //Reset row if value exceeds max rows, and reset
screen.
        row = 0;
        lcd.empty();
      }
    } else {
      if (curWord.length() == 1) {
         lcd.at(row, col, curWord.substring(0, 1) + " "); //Display the character
of the string plus a space.
         col += 2;
      } else {
         lcd.at(row, col, curWord.substring(0, 1)); //Display the character of the
string.
         col++;
      }
      curWord = curWord.substring(1); //Shorten the string.
    }

  }

}
```

- It worked, and I'm happy.

Note: Make sure to NEVER use the delay() function, as that will prevent the Arduino from reading any inputs. No corner counting.

Testing the Ability of the Arduino to Read Input

Reading a Switch

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
ParallaxLCD lcd(2,4,20);
const int BTN = 5; //Button Input
const int LED = 7; //Visual Evidence
```

```
void setup() {
  Serial.begin(9600);
  pinMode(BTN, INPUT);
  pinMode(LED, OUTPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (digitalRead(BTN)) { //If button pressed, send "ON" to display, send "ON" to
serial output, turn on LED.
    lcd.at(1,3, "ON ");
    Serial.println("ON");
    digitalWrite(LED, HIGH);
  } else { //If button not pressed, send "OFF" to display, send "OFF" to serial
output, turn off LED.
    lcd.at(1,3, "OFF");
    Serial.println("OFF");
    digitalWrite(LED, LOW);
  }

}
```

- Worked as intended. The Arduino is able to react as needed.

Analog Read

- What happens when analogRead() is being used. Will the output be voltage or a floating point number?
- Testing analogRead():

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
ParallaxLCD lcd(2,4,20);
const int SENSOR = A5; //Sensor Input (Photoresistor)
float x = 0.0;

void setup() {
  Serial.begin(9600);
  pinMode(SENSOR, INPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  x = analogRead(SENSOR); //Value of reading the sensor.
```

```
  Serial.println(x); //Print the value read at the sensor.
  lcd.at(1,3,String(x)); //Send the value read at the sensor to the LCD.

}
```

- The value being displayed was not an actual voltage, as its lowest value was still well above the numerical value of the Arduino voltage rating (5V).
- I used a voltage divider setup with one resistor and a potentiometer in series, the made the test pin electrically common with the joint between the resistor and potentiometer.
- By swapping input polarities for the voltage divider, and using the principles for how a voltage divider works, I found that the Arduino must use the ground pin as its reference. This means the component(s) between the wire electrically common to ground (GND) and the wire connecting to the analog pin (A5, in this case) is(are) or the components being measured. As for what the exact value being measured is, I do not know. However, I know how the referencing works now.

Note: The Arduino uses the ground pin (GND) as its reference point for measuring analog signals.

- Seems intuitive, but still helpful to know.
- The value was significant enough to notice without being digital ("ON" or "OFF"). This will be useful for detecting significant voltage for the photodiode circuit in the laser project.

Testing Readability Response While Controlling Other Things

- Flashed lights and tested LCD display and serial output updating.

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
ParallaxLCD lcd(2,4,20);
const int SENSOR = A5; //Sensor Input (Photoresistor)
float x = 0.0;

const int R_LED = 7;
const int G_LED = 5;
const int B_LED = 3;

void setup() {
  Serial.begin(9600);
  pinMode(SENSOR, INPUT);
  pinMode(R_LED, OUTPUT);
  pinMode(G_LED, OUTPUT);
  pinMode(B_LED, OUTPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {
```

```
  x = analogRead(SENSOR); //Value of reading the sensor.
  Serial.println(x); //Print the value read at the sensor.
  lcd.at(1,3,String(x)); //Send the value read at the sensor to the LCD.


  //Flashing Lights

  if (millis() % 20 == 0) {
    digitalWrite(R_LED, HIGH);
  }
  if (millis() % 20 == 10) {
    digitalWrite(R_LED, LOW);
  }

  if (millis() % 200 == 0) {
    digitalWrite(G_LED, HIGH);
  }
  if (millis() % 200 == 100) {
    digitalWrite(G_LED, LOW);
  }

  if (millis() % 2000 == 0) {
    digitalWrite(B_LED, HIGH);
  }
  if (millis() % 2000 == 1000) {
    digitalWrite(B_LED, LOW);
  }

}
```

- I saw no visual or recorded problem. The arduino can handle other processes while reading an input.

Testing Arduino Time Syncing

Syncing With a Sensor Input

- Turning on a light near a photoresistor connected in seris with a resistor, and measuring the value across the resistor.

Failed Attempt

```
#include <ParallaxLCD.h>

const int ROWS;
const int COLS;
const int LCDPinOut = 2;
ParallaxLCD lcd(LCDPinOut,ROWS,COLS);

const int BTN = A5;
String syncValue = "";
int syncTime = 0;
```

```
int newTime = 0;
int count = 0;
int x = 0;

void setup() {
  Serial.begin(9600); //Baud rate for input.
  pinMode(LCDPinOut, OUTPUT);
  pinMode(BTN, INPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  x = analogRead(BTN); //Assign sensor input to x.
  lcd.at(0,3,x); //Display sensor value.

  if (x > 900 && syncTime == 0) {
     syncValue = String(x); //The value on the sensor at sync time.
     syncTime = millis(); //The time it synced with input.
  }

  lcd.at(1,3,syncValue);

  newTime = millis() - syncTime; //Current time in mS since time sync.
  lcd.at(2,3, newTime % 10); //Remainder when dividing by 10.

  if (newTime % 10 == 0 && syncTime != 0) { //Add the number of times newTime was
divisible by 10.
    count++;
  }
  lcd.at(3,3,count); // Display the current count.
  lcd.at(3,10,(newTime/1000 * 100)); //Display the number of count that should
exist (should be 100 per second).

}
```

- Measuring negative values when taking modulous.
- newTime was negative.
- Not working properly.

Another Failed Attempt

```
#include <ParallaxLCD.h>

const int ROWS;
const int COLS;
const int LCDPinOut = 2;
ParallaxLCD lcd(LCDPinOut,ROWS,COLS);
```

```
const int BTN = A5;
String syncValue = "";
int syncTime = 0;
int newTime = 0;
int count = 0;
int x = 0;

void setup() {
  Serial.begin(9600); //Baud rate for input.
  pinMode(LCDPinOut, OUTPUT);
  pinMode(BTN, INPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  x = analogRead(BTN); //Assign sensor input to x.

  if (x > 900 && syncTime == 0) {
     syncValue = String(x); //The value on the sensor at sync time.
     syncTime = millis(); //The time it synced with input.
  }

  newTime = abs(millis() - syncTime); //Current differenc in time in mS since time
sync.
  if (newTime < 0) {
    newTime *= -1;
  }

  if (newTime % 10 == 0 && syncTime != 0) { //Add the number of times newTime was
divisible by 10.
    count++;
  }

  if (millis() % 100 == 0) {
    lcd.at(0,3,x); //Display sensor value.
    lcd.at(1,3,syncValue); //Display the sync value.
    lcd.at(3,3,count); // Display the current count.
    lcd.at(3,10,(newTime/1000 * 100)); //Display the number of count that should
exist (should be 100 per second).
    lcd.at(2,3, newTime % 10); //Remainder when dividing by 10.
    lcd.at(2,10,millis()); //Display current time in milliSeconds.
  }

}
```

- newTime still negative.
- Not working properly at all.
- Measuring the difference in predicted count versus the count value was inconsistent, and kept changing. Different rate of increase.

Measuring Accuracy of Using time for Syncing

```
#include <ParallaxLCD.h>

const int ROWS;
const int COLS;
const int LCDPinOut = 2;
ParallaxLCD lcd(LCDPinOut,ROWS,COLS);

double count = 0;

void setup() {
  pinMode(LCDPinOut, OUTPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (millis() % 10 == 0) { //Increases count every 10 milliSeconds.
    count++;
  }

  lcd.at(1,3, millis()/1000); //Time in milliSeconds since start.
  lcd.at(2,3, String(count)); //Current count value.
  lcd.at(3,3, String(10*count/(millis() / 1000))); //A way of measuring the
frequency in Hz of count increasing.

}
```

- Different rate of increase in time verus count.
- Count did not match what it should have.
- Frequency of count was not 100Hz, but kept fluctuating.

Note: Using the modulous operator (%) directly will not work for syncing time or displaying a value at the correct time.

Measuring Processing Time

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS);

long int startTime = 0;
long int endTime = 0;
long int difTime = 0;
```

```
void setup() {
  pinMode(LCDPin, OUTPUT);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (millis() < 2000) { //Clear the screen within 2 seconds after startup.
    lcd.empty();
  }

  startTime = millis(); //Started Processing


  //This is just meant for making the Arduino do something.
  for (int i = 0; i < 10; i++) {
    lcd.at(0,10,"A");
    lcd.at(0,10,"B");
    lcd.at(0,10,"C");
    lcd.at(0,10,"D");
    lcd.at(0,10,"E");
    lcd.at(0,10,"F");
    lcd.at(0,10,"G");
    lcd.at(0,10,"H");
    lcd.at(0,10,"I");
    lcd.at(0,10,"J");
    lcd.at(0,10,"K");
    lcd.at(0,10,"L");
    lcd.at(0,10,"M");
    lcd.at(0,10,"N");
    lcd.at(0,10,"O");
    lcd.at(0,10,"P");
    lcd.at(0,10,"Q");
    lcd.at(0,10,"R");
  }

  endTime = millis(); //Finished Processing

  difTime = endTime - startTime; //Processing Duration

  //Display the values.
  lcd.at(1,3, startTime);
  lcd.at(2,3, endTime);
  lcd.at(3,3, difTime);

}
```

- Using this code, the processing time was about 194.5 milliSeconds.
- I also tested this on an Arduino Nano board, and got the same exact number, 194-195 milliSeconds.

- Using a direct modulus function to find time to excute a function, or to use for time syncing, won't work.

Data Send and Recieve

Using Serial Data

First Trial

Code to Send a Message

```
void setup() {
  Serial.begin(9600);
  delay(1000);
}

void loop() {
  if (millis() < 2000) { //Before two seconds, do this, then, do that.
    Serial.println("EMPTY"); //Send a message.
  } else {
    Serial.println("NOT EMPTY");
  }
}
```

- Displayed "EMPTY" before 2000mS and "NOT EMPTY" after 2000mS, as expected.

Code to Recieve a Message

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); //LCD used for visual values.

String info = "";

void setup() {
  Serial.begin(9600);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (Serial.available()) { //If data comming in.
    info = Serial.readString(); //Assign data to info.
    Serial.println(info); //Output data to serial monitor.
```

```
  }

  lcd.at(1,3,String(millis())); //Display time since start in milliSeconds.

}
```

- The Arduino would stop function after 1-2 seconds, as the value displayed on the LCD would not update. There must be an error. This might be due to information being lost and the Arduino being confused.
- Also, when the Arduino was receiving "EMPTY", it functioned and updated. However, when the Arduino was supposed to be receiving "NOT EMPTY", it paused and didn't update. When un-pluging the wire connecting the Arduinos, the receiving Arduino started functioning properly again.
- Also, sometimes the receiving Arduino would not want to receiving uploaded code to modify its behavior.

Trial After Changing Strings to Characters

Reciever Code

- Slight change (outputin info value), but mostly the same.

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); //LCD used for visual values.

String info = "null";

void setup() {
  Serial.begin(9600);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (Serial.available()) { //If data comming in.
    info = Serial.readString(); //Assign data to info.
    Serial.println(info); //Output data to serial monitor.
  }

  lcd.at(1,3,info); //Display the value of info.

}
```

"Sender" Code

- Tried outputing characters instead of whole strings.

```
void setup() {
  Serial.begin(9600);
  delay(1000);
}

void loop() {
  if (millis() < 2000) { //Before two seconds, do this, then, do that.
    Serial.print('E'); //Send a message.
  } else {
    Serial.print('N');
  }
}
```

- Worked slightly, then didn't work, then made no sense and continuously stopped the program of the other Arduino.

Just Send Characters and Just Read Directly

Sender Code

```
void setup() {
  Serial.begin(9600);
  delay(1000);
}

void loop() {
  if (millis() < 2000) { //Before two seconds, do this, then, do that.
    Serial.print('E'); //Send a message.
  } else {
    Serial.print('N');
  }
}
```

Reciever Code

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); //LCD used for visual values.

String info = "null";
```

```
void setup() {
  Serial.begin(9600);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    Serial.println(info); //Output data to serial monitor.
  }

  lcd.at(1,3,info); //Display value of info.

}
```

- It worked!
- The information was stable. The reason it might have been having trouble is that the limit on package size is about 5-9 bits, and 8 bits are usually used per character, which a whole string of text is longer than. (Obviously, but still.)
- The only downside is that the data was in the form of a decimal number representing the character in ASCII. However, this isn't terrible, and is totally workable.

Updating Reciever Code

- Casting info as a character.

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); //LCD used for visual values.

int info = 0;

void setup() {
  Serial.begin(9600);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
```

```
    Serial.println(info); //Output data to serial monitor.
  }

  lcd.at(1,3,info); //Display value of info.
  lcd.at(2,3, (char) info); //Display the character coersponding with the ASCII
character numebr.

}
```

- I saw a character value as well as the ASCII numerical value, so it worked.

Testing Full Sentences

Character Array

Reciever

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); //LCD used for visual values.

int info = 0;

void setup() {
  Serial.begin(9600);
  lcd.setup();
  delay(1000);
  lcd.empty();
}

void loop() {

  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    Serial.println((char) info); //Output info as character to serial monitor.
  }

  lcd.at(1,3,String(info) + "       "); //Display value of info.
  lcd.at(2,3, (char) info); //Display the character coersponding with the ASCII
character numebr.

}
```

Sender

```
String text = "This is a sentence. I hope it reads properly. Testing a theory I
have.";
char letters[20];
int index = 0;
int endLength = text.length(); //Used later.

void setup() {
  Serial.begin(9600);
  delay(1000);
  text.substring(0, 20).toCharArray(letters, 20); //Take the string from index 0
to index 19 (20-1) and assign the characters in it to letters.
  text = text.substring(20); //Shorten text to only include text from index 20 and
on.
}

void loop() {
  if (millis() > 2000 && index < endLength) { //When index < endLength, text
length is < 20, and because of later code it will be empty. In short, it reached
the end of the sentence.
    Serial.write(letters[index]);
    index++; //Advance index in character array.
    if (index >= sizeof(letters)) { //Finished writing from character array.
      index = 0;
      if (text.length() >= 20) { //Update character array and text contents.
        text.substring(0, 20).toCharArray(letters, 20);
        text = text.substring(20);
      } else { //Update character array and set text contents to nothing, while
accounting for the "endLength" of the text.
        text.toCharArray(letters, 20);
        endLength = text.length();
        text = "";
      }
    }
    delay(10);
  }
}
```

- Several characters were excluded.

Using charAt() Function

- Reciever code the same as the pervious test (character array).

Sender

```
String text = "This is a sentence. I hope it reads properly. Testing a theory I
have.";

void setup() {
  Serial.begin(9600);
```

```
  delay(1000);
}

void loop() {
  if (millis() > 2000 && text.length() > 0) {
    Serial.write(text.charAt(0)); //Write the character at the beginning of the
text string.
    text = text.substring(1); //Remove the first character from string.
    delay(10);
  }
}
```

- Effective and simple.

Simulated Function

Sender Code

```
String text = "This is a sentence. I hope it reads properly. Testing a theory I
have. We'll see how well this works.";

void setup() {
  Serial.begin(9600); //Baud rate.
  delay(1000);
}

void loop() {
  if (millis() > 2000 && text.length() > 0) { //Wait until 2 seconds have passed
since startup, and make sure that there are characters remaining in "text".
    Serial.write(text.charAt(0)); //Write the character from the beginning of the
text string.
    text = text.substring(1); //Remove the first character from string. Also could
be thought of as only keeping the letters past the first one.
    delay(10); //This is to prevent it from being "instant".
  }
}
```

Reciever Code

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); // desired pin, rows, cols

String text = ""; //Captures the incoming text from the serial.
String curWord = ""; //Stores a single word at a time for displaying to the LCD.
```

```
int spaceIndex = 0; //A number used to modify a conditional statement, as
demonstrated in updateCursor().
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

int info = 0;
int dIndex = -1;
int sIndex = -1;
int spaceModifier = 0;

void setup () {
  Serial.begin(9600);
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  readInput();
  processText();

}

//Scan for incoming information.
void readInput() {
  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    text += String((char) info);
    Serial.println(text); //Used for diagnostics.
  } else {
    info = 0;
  }
}

//Processing data and displaying it on the screen.
void processText() {

  if (curWord.length() < 1) { //If current word is empty, then update it.
    if (text.length() > 0) { //Make sure there is actually data to transfer.
      dIndex = text.indexOf("."); //Does it have a period and where is it at?
      sIndex = text.indexOf(" "); //Does it have a space and where is it at?
      if (sIndex > -1) { //If space...
        curWord = text.substring(0, sIndex + 1); //Set curWord to the word
seperated by the space.
        text = text.substring(sIndex+1); //Take the current word out of text.
        spaceModifier = 1;
      } else if (dIndex > -1) {
        curWord = text.substring(0, dIndex + 1);
        text = text.substring(dIndex+1);
        spaceModifier = 0;
      }
      updateCursor();
    }
```

```
  } else { //If the current word has any characters, display them.
    disChar();
  }

}

//Update the Cursor
void updateCursor() {
  if (curWord.length() + col > COLS + spaceModifier) { //If word too long, wrap,
but don't include the space as part of the word if it exceeds the screen.
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
  }
}

//Display a Character
void disChar() {
  lcd.at(row, col, curWord.substring(0,1)); //Display a character at the current
cursor position.
  if (curWord.length() > 1) { //Default shortening.
    curWord = curWord.substring(1);
  } else {
    curWord = ""; //.substring() can't be used to shorten a letter, so instead,
set it to nothing.
  }
  col++; //Increase the cursor's column position.
}
```

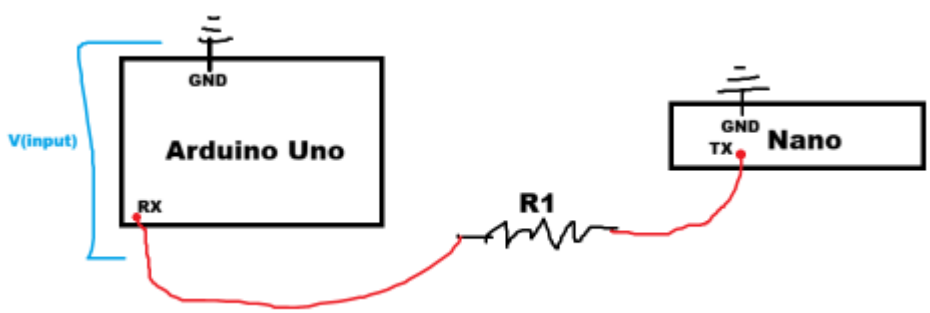
- After some modification to get to this code, it worked and it worked well. Text was displayed to the screen in a nice and orderly format. All characters were transmitted and displayed properly.

Signal Testing & Laser Light Communication

Experimentation

Testing the Effects on Arduino Reading when Signal Voltage is Modified

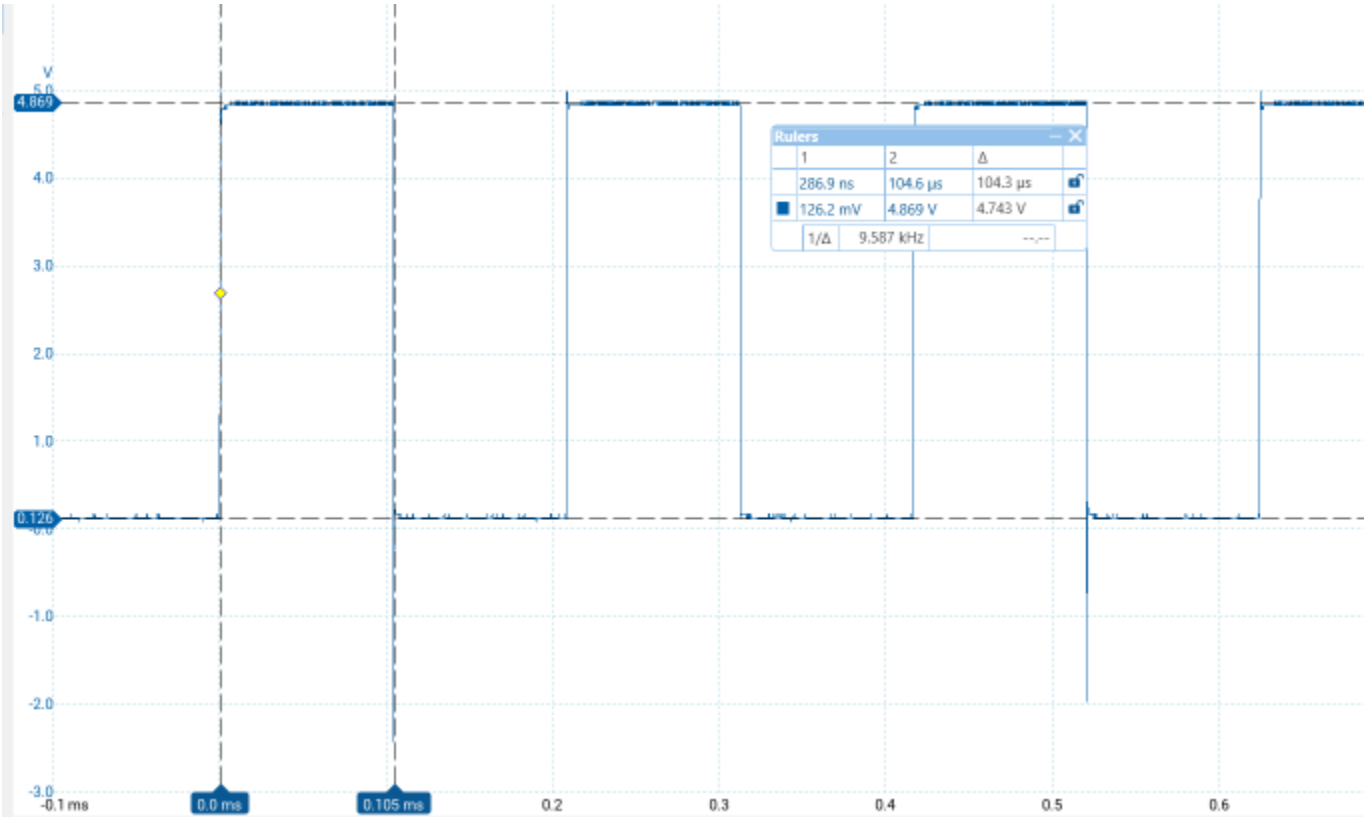
Method



- Use code for basic sentence display (as of 10/17/2025, same as in "Code Trials" documentation). This code basically sends serial charaters from one Arduino to be processed and displayed at another.
- Change resitance value of R1.
- Measure voltage at the input of the Arduino Uno with reference to ground, measuring voltage peaks and troughs.

Results

PicoScope Example



- Using the 0.0Ω resistor (wire).
- This is an example of the binary serial data being transmitted, with a value near 0V representing LOW (or a 0) and a value near 5V representing HIGH (or a 1).

Data Collection

Voltage Peak	4.943	4.943	4.943	4.943	4.943	4.862	4.943	4.943	4.943	4.943	4.943	V
Voltage Trough	0.1235	0.978	1.3	2.233	2.291	3.236	3.351	3.593	3.662	4.111	4.48	V
R1 Resistance	0.0	220.2	328.9	471.5	847	1.989k	2.132k	2.661k	2.944k	5.07k	9.97k	Ω
Readable (Y/N)	Y	Y	Y	Y	N	N	N	N	N	N	N	
				p	p		p	p	p			

- Peak is the maximum input voltage reached during transmission.
- Trough is the minimum input voltage reached during transmission.
- The resistance is of the connecting data wire, from the Nano's TX (output) pin to the Uno's RX (input) pin.
- Any column with a "p" at the bottom means a potentiometer was used as that resistor, all other values are from fixed resistors.

Other Notes

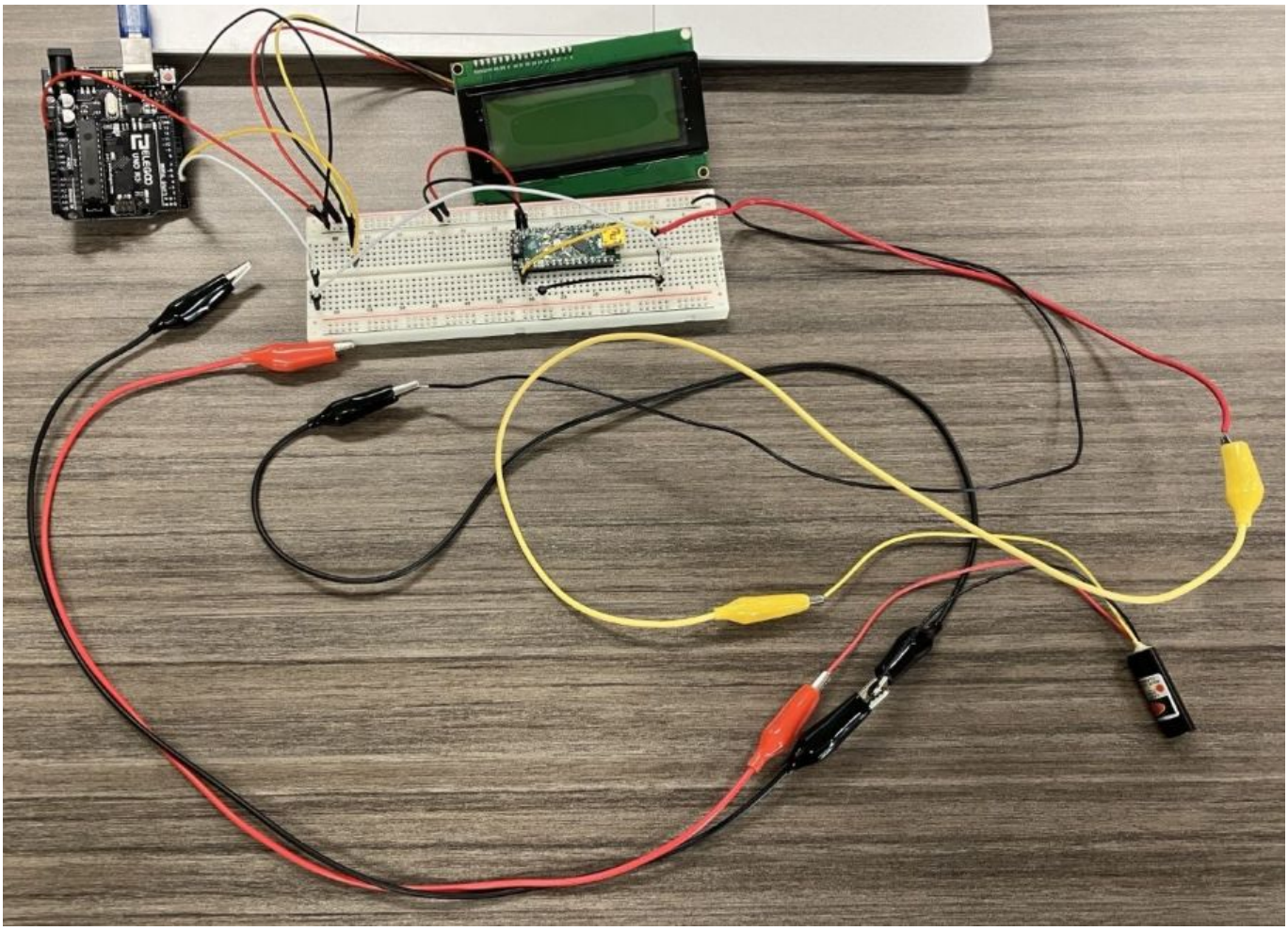
- One bit lasts about 104.3µS, with about 9.26mS in-between each packet (set of bits).
- When connecting the RX pin to ground on the arduino uno, I get about 4.75mA to 4.78mA of current. 4.88 to 4.92V across the pins.
- When connecting the TX pin to ground on the arduino uno, I get about 82.4mA to 82.8mA of current. 4.91 to 4.92V across the pins.
- When connecting the TX pin to ground on the arduino nano, I get about 75.9mA to 76.0mA of current. 4.92V across the pins.
- When connecting the RX pin to ground on the arduino nano, I get about 3.66mA to 3.67mA of current. 4.92V across the pins.
- 0.30mA max current when using mulitmer min/max function and connecting the multimeter along the data wire.
- Interestingly, the RX and TX pins on the Arduinos are acting more like sources or like they have internal pull-up resistors, rather than directly being digital logic pins. Although, this makes sense, as the RX and TX pins are serial data pins and need to be able to switch to on and off states very quickly.

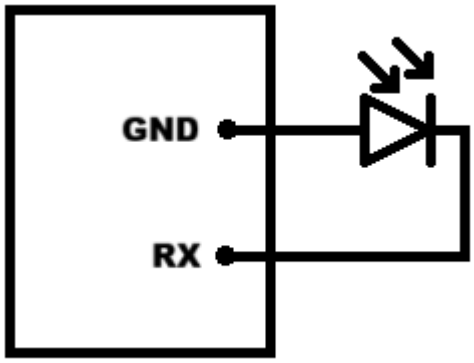
Laser Light Communication Experimentation

Direct Connection

Setup

- This was a test of connecting the photodiode directly between the RX and ground pin, and observing the results.





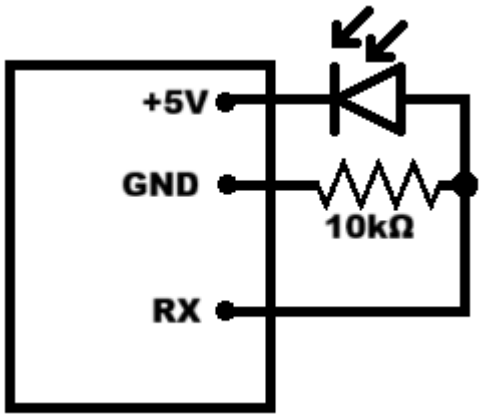
(Reciever Board)

Results

- When the laser was not striking it, I got about 5V across the photodiode, and the voltage only dropped about 1V when the laser was striking the diode. Directly connecing the photodiode will not work.
- 5V when at rest and 4.5 Volts (from RX to GND) when stimulated.

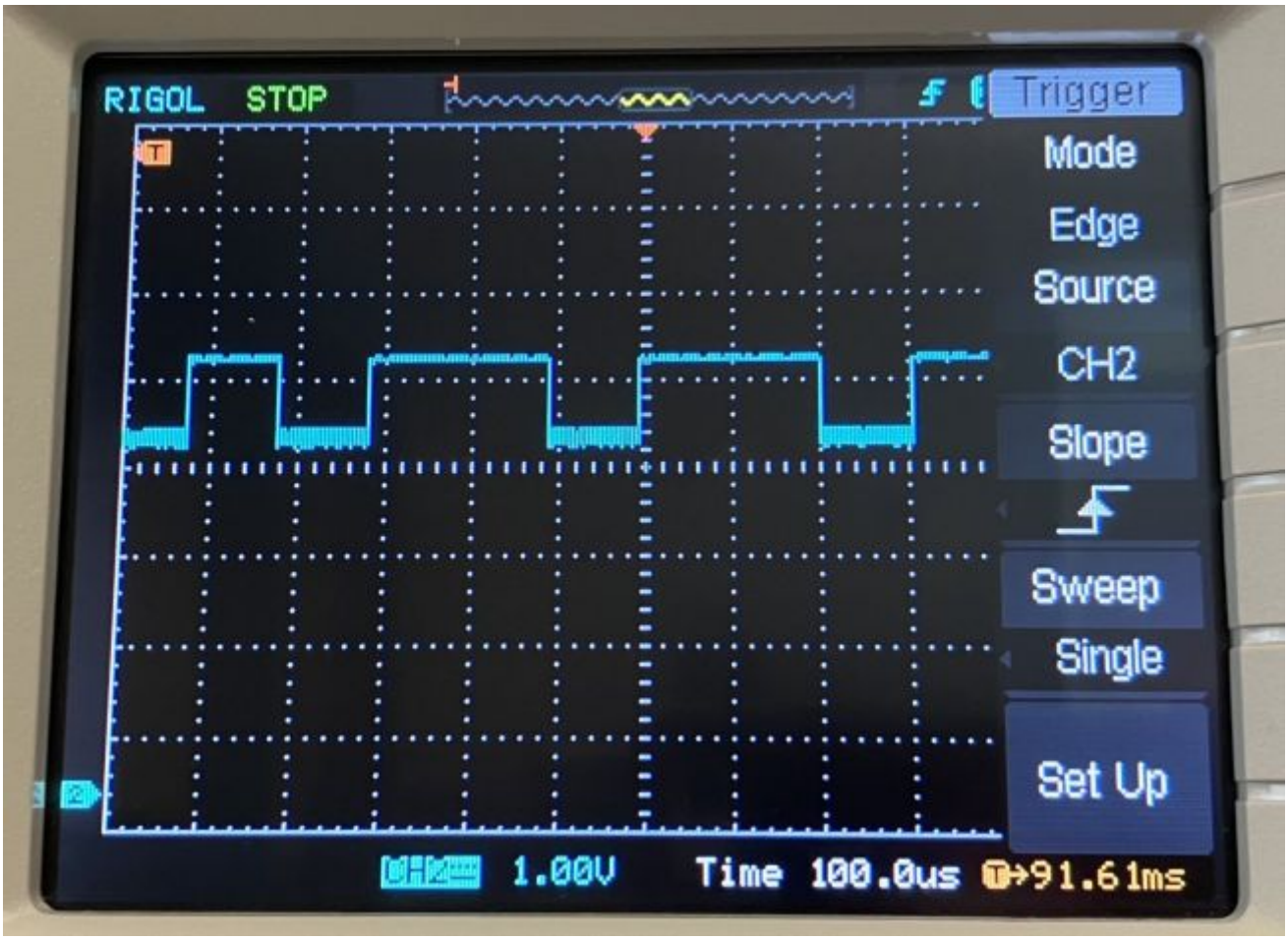
Resistor-Photodiode Pair

Resistor on Ground Side



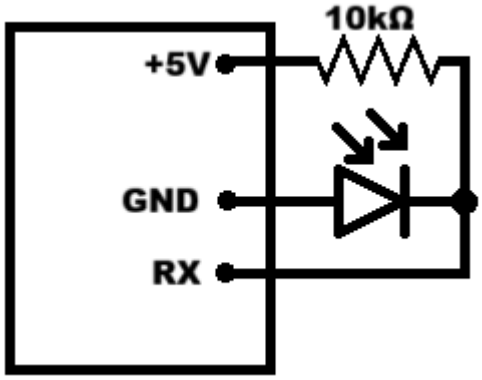
(Reciever Board)

- Peak voltage of 5V (un-stimulated).
- Trough voltage of 4V (stimulated).
- Voltage from RX pin to GND.



- Didn't work to bring the voltage down to zero.

Photodiode on Ground Side

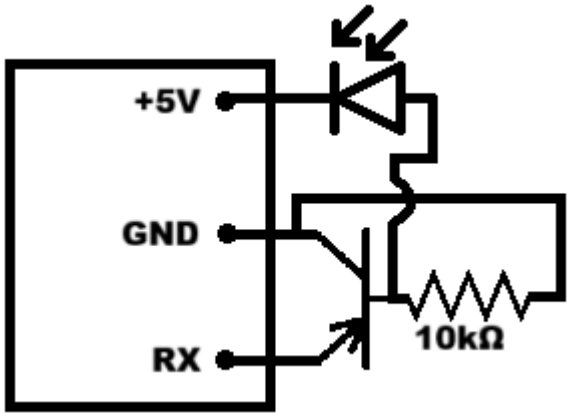


(Reciever Board)

- Peak voltage of 4.5V (un-stimulated).
- Trough voltage of 4V (stimulated).
- Voltage from RX pin to GND.

Semi-Functioning Attempt

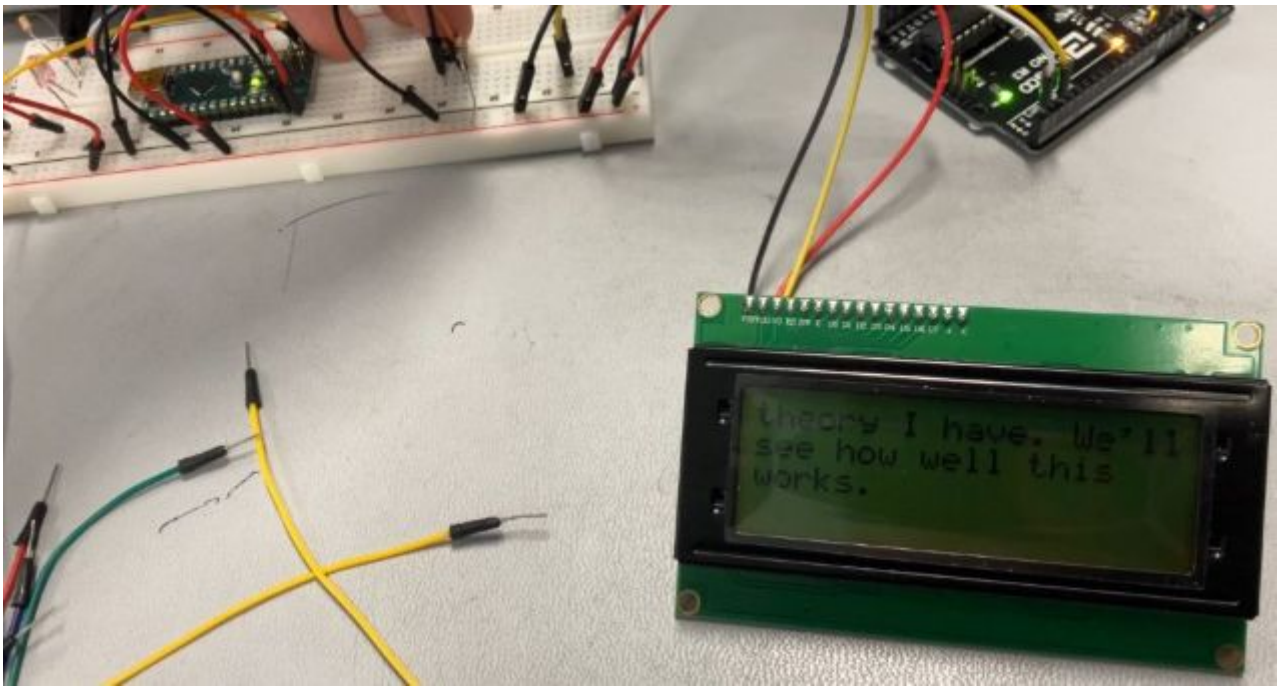
Setup



(Reciever Board)

- Intended to make RX HIGH when no light was hitting it and LOW otherwise.
- As to why I built the circuit like this, I am not fully sure. However, it somewhat worked.

Running

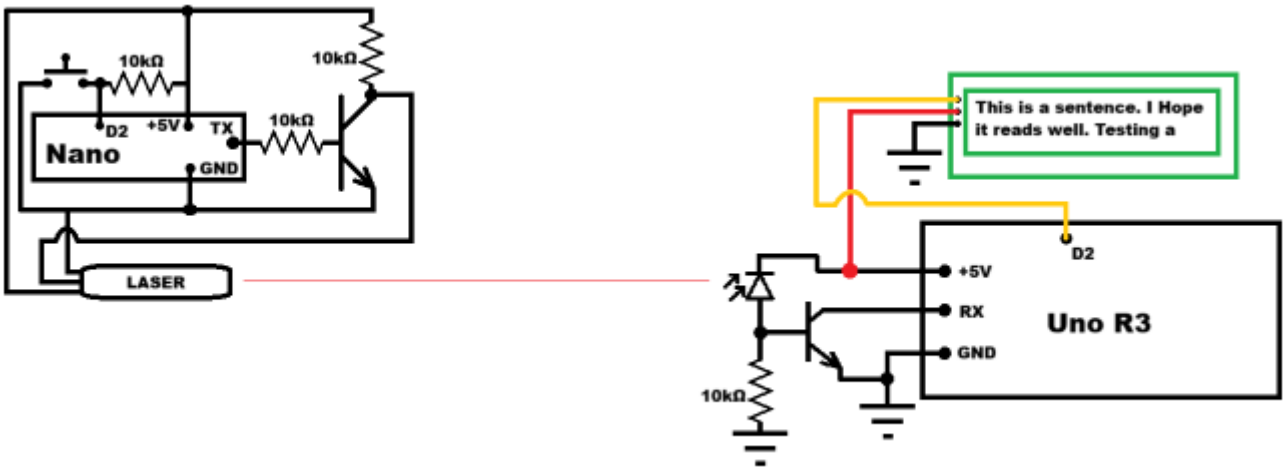


Results

- The screen would sometimes fill up with all the characters correctly, which means the method was working at times. However, there were several times that letters were missing (sometimes few, sometimes a lot), and other times where nothing showed up at all or the buzzer on the back of the LCD made noise.
- I thougths that part of the reason the serial data may not be communicating properly could also be due to my code setup, as I had decided to make the Nano (serial sending Arduino) display serial data every time it restarted. However, when the Nano is restarting, it turns of the serial pin, which is normally one. This pin turns on and off at least two times before sending the data. It is possible that there is not a long enough time for the Arduino Uno (receiving end) to recognize the error. However, after making a button display the text again without resetting the entire Nano's runtime, it still had issues displaying properly on the Uno R3's side.
- Since this was not consistent, it is not functional, and so I determined to find a better way to control the laser and photodiode circuits.

Functional

Setup



- After some looking around at transistors, and with some help from Tony, I found a setup that worked properly for displaying data.
- I determined that I wanted the laser to not light when TX is HIGH, and for it to light up when RX is LOW, which is how the laser circuit functioned.
- Because of the way the serial transmitting works and the laser cirucit setup, I needed RX to be HIGH when the photodiode was not being stimulated and vice-versa.
- With my original idea for the receiving (photodiode) circuit, the circuit was doing the oppostie effect as I wanted, causing it to be LOW when the photodiode was not stimulated. However, by simply swapping the 10k resistor with the photodiode (as is currently shown in the schematic on the "Uno R3" side), the circuit worked properly, inverting the laser signal.

Results

- Text displayed!
- Worked every time.
- Worked over some distance and transmitted far more effectively than any previous versions.
- No weird sounds (LCD buzzer) or faulty display on the receiving end.
- Very functional... now to test over distances.

One Caveat

- At one point, the laser light was very dim, and it never got brighter. This prevented me from doing further tests with the system I had set up.
- I tested the laser using a power supply instead of the Arduino board, and it was still quite dull.
- I took a current measurement while the laser was powered and got a current of about 126.1mA. The datasheet for the laser said it was rated for a max current of 35mA, so uh... oops.
- The laser might need to have a resistor on one of its power inputs to prevent overcurrent (using the other laser).
- However, I am quite curious as to why it just today decided to go out, as I had several other times where it was running directly off 5V, with resistor limiting current. Also, the laser was on for a much longer period of time during those other tests, or at least, so I thought.
- Might it have something to do with a loading effect on the Arduino (the laser was powered by the Uno R3 at the time of dimming) causing excess voltage across the laser? Might the laser have overheated and failed a component? Might the time I felt a static shock have damaged the laser (this was on the breadboard, but still)?

- I have two main suspicions on why the laser failed. One, and most likely, is that the laser was receiving way too much current. Two, while unlikely, is that the static discharge could have affected the Nano (and indirectly the laser) or the laser itself, causing a failed internal component. This second one doesn't seem as likely, but I remember the static discharge wasn't too long before the laser failed.
- In short, the laser failed. The system works, but I need to be smarter about handling the laser.

Note: Laser current is a good thing to regulate, as well as protection from static discharge.

Current and Voltage Measurements

- All tests where done with a ~5.02V source (Uno R3 Board)
- The laser was connected between +V (red wire) and GND (black wire), with the laser input (yellow wire) conneecting to +V (5V input), unless otherwise specified.

Broken Laser

- Along black wire with yellow wire disconnected and red connected: 0.95mA
- Along black wire with yellow wire connected and red disconnected: 1.43mA
- Along red wire with black disconnected and yellow connected: 0.0mA
- Along red wire, all wires connected: 127.9mA (started at about 126mA)
- Along black wire, all wires connected: 128.0mA (+/- 0.2mA) (started at about 126mA)
- Along yellow wire, all wires connected: 1.31mA
- Along yellow wire, red connected to GND and yellow connected to +V (black disconnected): 0.71mA
- Along yellow wire, red connected to +V and yellow connected to GND (black disconnected):-127.2mA
- Voltage Full: 4.75V
- Voltage Red+Black: 5.03V
- Voltage Yellow+Black: 5.03V
- Voltage Red+Yellow: 5.03V (later 4.95V)
- The laser acted as a short when first plugged in with all wire, before returning to normal.

Functional Laser

- Along black wire with yellow wire disconnected and red connected: 0.97mA
- Along black wire with yellow wire connected and red disconnected: 1.40mA
- Along red wire with black disconnected and yellow connected: 0.0mA
- Along red wire, all wires connected: 12.83mA
- Along black wire, all wires connected: 14.27mA

- Along yellow wire, all wires connected: 1.42mA
- Along yellow wire, red connected to GND and yellow connected to +V (black disconnected): 0.68mA
- Along yellow wire, red connected to +V and yellow connected to GND (black disconnected): -0.35mA
- Voltage Full: 4.99V
- Voltage Red+Black: 5.02V
- Voltage Yellow+Black: 5.01V
- Voltage Red+Yellow: 5.01V

Most Likely Cause of Laser Failure:

- Static discharge.

Code Used for the Semi-Functional (and non-functional) Experimental Runs

Sender

```
String text = "This is a sentence. I hope it reads properly. Testing a theory I
have. We'll see how well this works.";

void setup() {
  Serial.begin(9600); //Baud rate.
  delay(1000);
}

void loop() {
  if (millis() > 2000 && text.length() > 0) { //Wait until 2 seconds have passed
since startup, and make sure that there are characters remaining in "text".
    Serial.write(text.charAt(0)); //Write the character from the beginning of the
text string.
    text = text.substring(1); //Remove the first character from string. Also could
be thought of as only keeping the letters past the first one.
    delay(10); //This is to prevent it from being "instant".
  }
}
```

- Waits to display based off time passed and after resetting the board.

Reciever

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
```

```
ParallaxLCD lcd(LCDPin, ROWS, COLS); // desired pin, rows, cols

String text = ""; //Captures the incoming text from the serial.
String curWord = ""; //Stores a single word at a time for displaying to the LCD.
int spaceIndex = 0; //A number used to modify a conditional statement, as
demonstrated in updateCursor().
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

int info = 0;
int dIndex = -1;
int sIndex = -1;
int spaceModifier = 0;

void setup () {
  Serial.begin(9600);
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  readInput();
  processText();

}

//Scan for incoming information.
void readInput() {
  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    text += String((char) info);
    Serial.println(text); //Used for diagnostics.
  } else {
    info = 0;
  }
}

//Processing data and displaying it on the screen.
void processText() {

  if (curWord.length() < 1) { //If current word is empty, then update it.
    if (text.length() > 0) { //Make sure there is actually data to transfer.
      dIndex = text.indexOf("."); //Does it have a period and where is it at?
      sIndex = text.indexOf(" "); //Does it have a space and where is it at?
      if (sIndex > -1) { //If space...
        curWord = text.substring(0, sIndex + 1); //Set curWord to the word
seperated by the space.
        text = text.substring(sIndex+1); //Take the current word out of text.
        spaceModifier = 1;
      } else if (dIndex > -1) {
        curWord = text.substring(0, dIndex + 1);
        text = text.substring(dIndex+1);
```

```
        spaceModifier = 0;
      }
      updateCursor();
    }
  } else { //If the current word has any characters, display them.
    disChar();
  }

}

//Update the Cursor
void updateCursor() {
  if (curWord.length() + col > COLS + spaceModifier) { //If word too long, wrap,
but don't include the space as part of the word if it exceeds the screen.
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
  }
}

//Display a Character
void disChar() {
  lcd.at(row, col, curWord.substring(0,1)); //Display a character at the current
cursor position.
  if (curWord.length() > 1) { //Default shortening.
    curWord = curWord.substring(1);
  } else {
    curWord = ""; //.substring() can't be used to shorten a letter, so instead,
set it to nothing.
  }
  col++; //Increase the cursor's column position.
}
```

- (Same as functional.)

Code Used for the Functional Experimental Run

Sender

```
String text = "This is a sentence. I hope it reads properly. Testing a theory I
have. We'll see how well this works.";
int BTN = 2; //Used for resetting the serial output.
boolean done = true; //When all characters have been removed from text.

void setup() {
  Serial.begin(9600); //Baud rate.
  pinMode(BTN, INPUT); //Button as an input.
  delay(1000);
```

```
}

void loop() {
  if (digitalRead(BTN) != HIGH) { //Yes, I could have said if "LOW", but it works.
    done = false; //You can't be finished if you're starting over.
    text = "This is a sentence. I hope it reads properly. Testing a theory I have.
We'll see how well this works."; //"Refill" text.
  } else {
    if (!done) { //Display the words within text, but only after button is
released.
      disWords();
    }
  }
}

void disWords() {
  if (text.length() > 0) { //Wait until 2 seconds have passed since startup, and
make sure that there are characters remaining in "text".
    Serial.write(text.charAt(0)); //Write the character from the beginning of the
text string.
    text = text.substring(1); //Remove the first character from string. Also could
be thought of as only keeping the letters past the first one.
    delay(10); //This is to prevent it from being "instant".
  } else {
    done = true; // If there are no more characters in text, then it has finished.
  }
}
```

- Re-displays the text after a button press, rather than resetting the whole board.

Reciever

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); // desired pin, rows, cols

String text = ""; //Captures the incoming text from the serial.
String curWord = ""; //Stores a single word at a time for displaying to the LCD.
int spaceIndex = 0; //A number used to modify a conditional statement, as
demonstrated in updateCursor().
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

int info = 0;
int dIndex = -1;
int sIndex = -1;
int spaceModifier = 0;
```

```
void setup () {
  Serial.begin(9600);
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  readInput();
  processText();

}

//Scan for incoming information.
void readInput() {
  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    text += String((char) info);
    Serial.println(text); //Used for diagnostics.
  } else {
    info = 0;
  }
}

//Processing data and displaying it on the screen.
void processText() {

  if (curWord.length() < 1) { //If current word is empty, then update it.
    if (text.length() > 0) { //Make sure there is actually data to transfer.
      dIndex = text.indexOf("."); //Does it have a period and where is it at?
      sIndex = text.indexOf(" "); //Does it have a space and where is it at?
      if (sIndex > -1) { //If space...
        curWord = text.substring(0, sIndex + 1); //Set curWord to the word
seperated by the space.
        text = text.substring(sIndex+1); //Take the current word out of text.
        spaceModifier = 1;
      } else if (dIndex > -1) {
        curWord = text.substring(0, dIndex + 1);
        text = text.substring(dIndex+1);
        spaceModifier = 0;
      }
      updateCursor();
    }
  } else { //If the current word has any characters, display them.
    disChar();
  }

}

//Update the Cursor
void updateCursor() {
  if (curWord.length() + col > COLS + spaceModifier) { //If word too long, wrap,
but don't include the space as part of the word if it exceeds the screen.
```

```
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
  }
}

//Display a Character
void disChar() {
  lcd.at(row, col, curWord.substring(0,1)); //Display a character at the current
cursor position.
  if (curWord.length() > 1) { //Default shortening.
    curWord = curWord.substring(1);
  } else {
    curWord = ""; //.substring() can't be used to shorten a letter, so instead,
set it to nothing.
  }
  col++; //Increase the cursor's column position.
}
```

Keyboard Input

Experimentation

Keyboard Experimentation

Pre-implentation Notation

Before I can experiment with the keyboard, I need the display method to be updated. Currently, the Arduino Uno (receiving end) is displaying the characters as whole words at a time. I need it to display characters as they come and not whole words.

Goal:

- Display the character as it's recieved.
- Once the word exceeds the screen, replace it with space and move it to a new line.

Adjusting the Display Code

Sending Code (From Arduino Due)

```
String text = "This is a sentence. I hope it reads properly. Testing a theory I
have. We'll see how well this works.";
int BTN = 2; //Used for resetting the serial output.
boolean done = true; //When all characters have been removed from text.

void setup() {
  Serial.begin(9600); //Baud rate.
```

```
  pinMode(BTN, INPUT); //Button as an input.
  delay(1000);
}

void loop() {
  if (digitalRead(BTN) != HIGH) { //Yes, I could have said if "LOW", but it works.
    done = false; //You can't be finished if you're starting over.
    text = "This is a sentence. I hope it reads properly. Testing a theory I have.
We'll see how well this works."; //"Refill" text.
  } else {
    if (!done) { //Display the words within text, but only after button is
released.
      disWords();
    }
  }
}

void disWords() {
  if (text.length() > 0) { //Wait until 2 seconds have passed since startup, and
make sure that there are characters remaining in "text".
    Serial.write(text.charAt(0)); //Write the character from the beginning of the
text string.
    text = text.substring(1); //Remove the first character from string. Also could
be thought of as only keeping the letters past the first one.
    delay(50); //This is to prevent it from being "instant".
  } else {
    done = true; // If there are no more characters in text, then it has finished.
  }
}
```

- The only difference from the previous version is that the delay between characters is set to 50 milliSeconds, rather than 10mS, so that I have more time to observe.

Failed Reciever End (Arduino Uno)

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); // desired pin, rows, cols

String text = ""; //Captures the incoming text from the serial.
String curWord = ""; //Stores the current word that may or may not exceed the
line.
String curChar = ""; //Use to display the current character.
String spaces = ""; //Used to clear word.
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

int info = 0; //Store incoming data.
```

```
int spaceR; //Row position of last space.
int spaceC; // Column position of last space.

void setup () {
  Serial.begin(9600);
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  readInput();
  processText();

}

//Scan for incoming information.
void readInput() {
  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    text += String((char) info);
    Serial.println(text); //Used for diagnostics.
  } else {
    info = 0;
  }
}

//Processing data and displaying it on the screen.
void processText() {

  if (text.length() > 0) { //Make sure there is actually data to transfer.

    curChar = text.substring(0,1); //Add the first character from the serial to
the current character.
    curWord += curChar; // Add the current character to current word.
    text = text.substring(1); //Shorten text.
    if (curChar.equals(" ")) {
      spaceR = row; //Row index of the previous space.
      spaceC = col; //Column index of the previous space.
    }
    disChar();
    updateCursor();
  }
}

//Update the Cursor
void updateCursor() {
  if (col > COLS - 1) { //If word too long, wrap.

    row = spaceR; //Start at the space index row.
    col = spaceC + 1; //Start just after space index column.
    for (int i = 1; i < curWord.length(); i++) { //Insert number of space require
to "erase" the word.
```

```
      spaces += " ";
    }
    lcd.at(row, col, spaces); //Erase the last word.
    spaces = "";
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
    lcd.at(row, col, curWord); //Print the word that was erased from the previous
row.
    curWord = "";
  }
}

//Display a Character
void disChar() {
  lcd.at(row, col, curChar); //Display a character at the current cursor position.
  curChar = "";
  col++; //Increase the cursor's column position.
}
```

- This didn't quite work right. The display was printing a whole sentence on the next line, but after going through and commenting the code, I realized why. The space index was updated to include the row and column index of the previous space, but it wasn't erasing the current word, so current word represented a whole line of text, instead of one word.

Next Attempt

- I added code to empty current word once the space index was establised, within the block that update space row and column index.

```
if (curChar.equals(" ")) {
    spaceR = row; //Row index of the previous space.
    spaceC = col; //Column index of the previous space.
    curWord = ""; //Clears the current word.
}
```

- This did prevent whole sentences from being displayed, but the display did not erase the last letter on the previous line and was missing between one and several characters on the next line.
- Using the serial monitor to display the text as it was recieved, the Arduino does know what it's being fed, so it is something to do with moving the text down a line.

Third Attempt

- Modifying the spaces function to not start at i=1, but rather i=0. This way the proper number of space will be added. I originally thought I would need one less space than the current word length, until I

looked over my code and realized I was wrong.

- Also, updating the character position to start at the end of the current word, instead of writing over it, which is why it appeared like characters were missing.

```
void updateCursor() {
  if (col > COLS - 1) { //If word too long, wrap.

    row = spaceR; //Start at the space index row.
    col = spaceC + 1; //Start just after space index column.
    for (int i = 0; i < curWord.length(); i++) { //Insert number of space require
to "erase" the word.
      spaces += " ";
    }
    lcd.at(row, col, spaces); //Erase the last word.
    spaces = "";
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
    lcd.at(row, col, curWord); //Print the word that was erased from the previous
row.
    row += curWord.length(); // Move cursor to past the end of the current word
segment.
    curWord = "";
  }
}
```

- Didn't erase the current word, but didn't display the whole word.
- Also, caused the lcd buzzer to go off.
- This is likely due to the row being updated by the shift, instead of the column. I don't know why I put "row" there.

Functional Attempt

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); // desired pin, rows, cols

String text = ""; //Captures the incoming text from the serial.
String curWord = ""; //Stores the current word that may or may not exceed the
line.
String curChar = ""; //Use to display the current character.
String spaces = ""; //Used to clear word.
```

```
int row = 0; //Start at row 0.
int col = 0; //Start at columm 0.

int info = 0; //Store incoming data.
int spaceR; //Row position of last space.
int spaceC; // Column position of last space.

void setup () {
  Serial.begin(9600);
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  readInput();
  processText();

}

//Scan for incoming information.
void readInput() {
  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    text += String((char) info);
    Serial.println(text); //Used for diagnostics.
  } else {
    info = 0;
  }
}

//Processing data and displaying it on the screen.
void processText() {

  if (text.length() > 0) { //Make sure there is actually data to transfer.

    curChar = text.substring(0,1); //Add the first character from the serial to
the current character.
    text = text.substring(1); //Shorten text.
    curWord += curChar; // Add the current character to current word.
    if (curChar.equals(" ")) {
      spaceR = row; //Row index of the previous space.
      spaceC = col; //Column index of the previous space.
      curWord = ""; //Clears the current word.
    }
    disChar();
    updateCursor();
  }
}

//Update the Cursor
void updateCursor() {
  if (col > COLS - 1) { //If word too long, wrap.
```

```
    row = spaceR; //Start at the space index row.
    col = spaceC + 1; //Start just after space index column.
    for (int i = 0; i < curWord.length(); i++) { //Insert number of space require
to "erase" the word.
      spaces += " ";
    }
    lcd.at(row, col, spaces); //Erase the last word.
    spaces = "";
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
    lcd.at(row, col, curWord); //Print the word that was erased from the previous
row.
    col += curWord.length(); // Move cursor to past the end of the current word
segment.
    if (col > COLS - 1) { //Will be used for later in case the word exceed the
whole line.
      row += col / 20; //Number of rows the text exceeded.
      col = (col + 1) % 20; //The remaining column index after the text has
spilled over.
      if (row > ROWS - 1) {
        lcd.empty();
        row = 0;
        col = 0;
      }
    }
    curWord = "";
  }
}

//Display a Character
void disChar() {
  lcd.at(row, col, curChar); //Display a character at the current cursor position.
  curChar = "";
  col++; //Increase the cursor's column position.
}
```

- Allowed the text to be displayed with one character at a time, and the text wrapped a whole word to the next line when it went past the screen.

Keyboard Input

Arduino USB Host Documentation Example

```
#include <KeyboardController.h>

// Initialize USB Controller
```

```
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  Serial.print("Pressed:  ");
  Serial.print(keyboard.getKey());
  Serial.println();
}
```

- Watching the output from the serial monitor, it appears like the characters are in ASCII format, as only numbers were displayed. Even numeral keys displayed as ASCII characters. I did notice, however, that the value for backspace, tab, caps, and the special function keys displayed as "0", which means the microcontroller did not recognize the key. Also, the left shift button worked to change the letter to capital, but the right shift button did not. Also, the stream was not continuous, but only displayed a new character as a key was pressed, though this makes sense when looking at the invoking event called "keyPressed()".

Using Functions from USB Host Documentation

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  Serial.print("Pressed:  ");
  Serial.print(keyboard.getKey());
  Serial.print("; ");
  Serial.print("Modifiers: ");
```

```
  Serial.print(keyboard.getModifiers());
  Serial.print("; ");
  Serial.print("Oem Code:  ");
  Serial.print(keyboard.getOemKey());
  Serial.println();
}

void keyReleased() {
  Serial.print("Released:  ");
  Serial.print(keyboard.getKey());
  Serial.println();
}
```

- The "keyReleased()" function happened after the key was released, and corresponded to the correct one.
- The keyboard modifiers for alt, control, and shift were non-zero values. However, keys like backspace were zero for both the keypressed and modifier, but not on the Oem code. This means I will have to use the Oem code to detect the backspace key, but it's better than nothing.

Sending Keystokes

Basic Text

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  if (key) { //As long as key isn't zero (a null character), it will print.
    Serial.write(key);
    key = 0;
  }
}
```

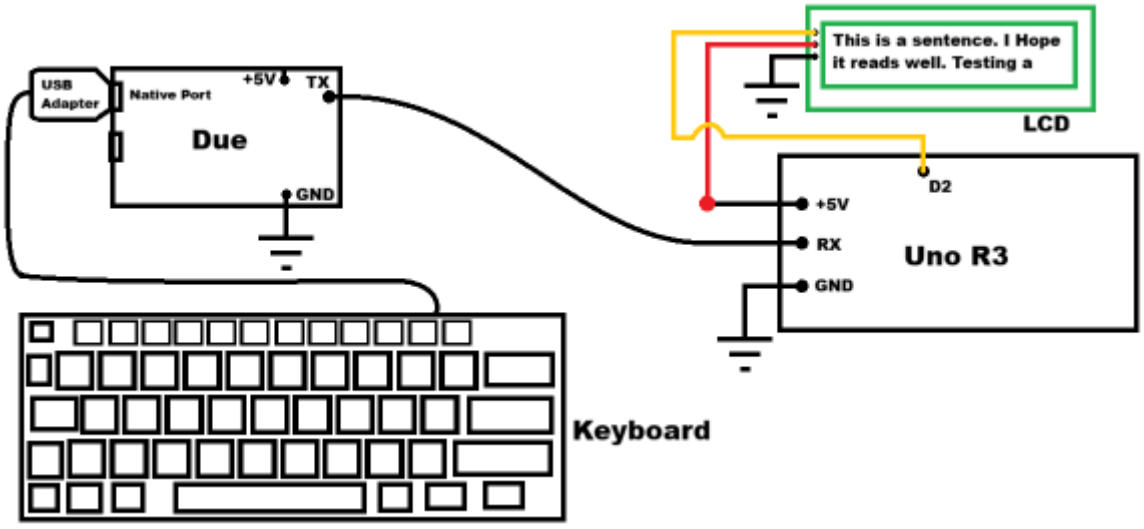
- Viewing from the serial monitor, it displayed the key as the actual character.
- Connecting it to the Arduino Uno, I see characters being displayed as I type them, and the words do wrap around. Now, I need to incorporate backspace as a functionality. I also noticed that when a really long text appeared, to where it passed the whole line, it started back at the line before writing. Earlier, I anticipated that this would happen, but now it needs to be accounted for. This is a receiving end issue, and has nothing to do with the Arduino Due. I also saw another inefficiency, which is that the text was wrapping before it needed to (by one column), but that is an error on the receiving end (Arduino Uno).
- The next day (10/29/2025), I tested the circuit using the laser as the communication method, and it worked.

Notes:

- **Make sure to have backspace function.**
- **Have a long string of text able to wrap correctly.**
- **Possibly add a scrolling functionality.**
- **Fix Arduino Uno pre-emptive wrap.**

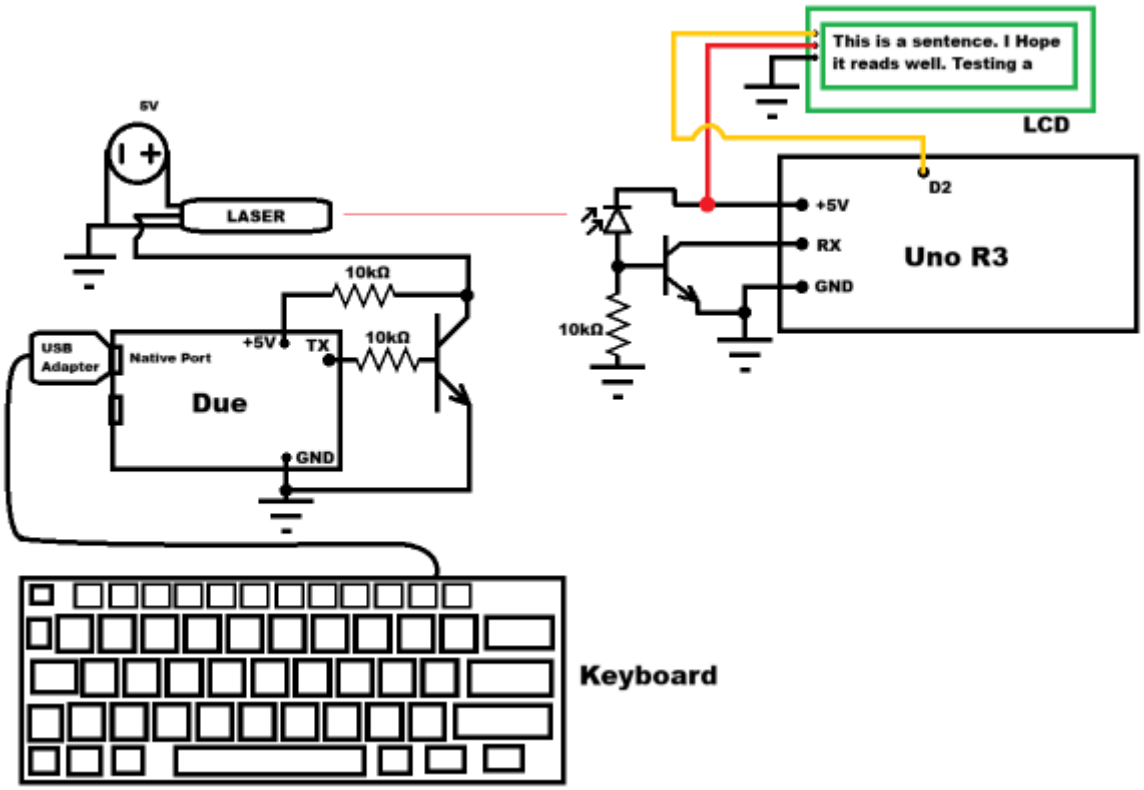
Test Setups

Setup For Keyboard Test Using Cable (Wire)



- This was done for all the experiment except testing with the laser. This was an easy way to test character sending without using the laser and photodiode.
- (Tested and experimented with on 10/28/2025)

Setup For Keyboard Test Using Laser and Photodiode



- This was only done once and shown as a functioning example.
- The code for the Arduino Due was the code found in the final result in the "Basic Text" section of "Sending Keystrokes". The Arduino Uno's code was that found in the "Functional Attempt" section of "Adjusting the Display Code".
- (Tested on 10/29/2025)

Optimizing Text Wrap

Experimentation

Optimizing text Wrapping

Starting Code

Uno R3

```
#include <ParallaxLCD.h>

const int ROWS = 4;
const int COLS = 20;
const int LCDPin = 2;
ParallaxLCD lcd(LCDPin, ROWS, COLS); // desired pin, rows, cols

String text = ""; //Captures the incoming text from the serial.
String curWord = ""; //Stores the current word that may or may not exceed the
line.
String curChar = ""; //Use to display the current character.
String spaces = ""; //Used to clear word.
int row = 0; //Start at row 0.
```

```
int col = 0; //Start at columm 0.

int info = 0; //Store incoming data.
int spaceR; //Row position of last space.
int spaceC; // Column position of last space.

void setup () {
  Serial.begin(9600);
  lcd.setup();
  delay(1000); // Wait for the LCD to start up.
  lcd.empty(); // Clear the screen.
}

void loop () {

  readInput();
  processText();

}

//Scan for incoming information.
void readInput() {
  if (Serial.available()) { //If data comming in.
    info = Serial.read(); //Assign data to info.
    text += String((char) info);
    Serial.println(text); //Used for diagnostics.
  } else {
    info = 0;
  }
}

//Processing data and displaying it on the screen.
void processText() {

  if (text.length() > 0) { //Make sure there is actually data to transfer.

    curChar = text.substring(0,1); //Add the first character from the serial to
the current character.
    text = text.substring(1); //Shorten text.
    curWord += curChar; // Add the current character to current word.
    if (curChar.equals(" ")) {
      spaceR = row; //Row index of the previous space.
      spaceC = col; //Column index of the previous space.
      curWord = ""; //Clears the current word.
    }
    disChar();
    updateCursor();
  }
}

//Update the Cursor
void updateCursor() {
  if (col > COLS - 1) { //If word too long, wrap.
```

```
    row = spaceR; //Start at the space index row.
    col = spaceC + 1; //Start just after space index column.
    for (int i = 0; i < curWord.length(); i++) { //Insert number of space require
to "erase" the word.
      spaces += " ";
    }
    lcd.at(row, col, spaces); //Erase the last word.
    spaces = "";
    col = 0; //Reset cursor at column 0.
    row++; //Move the cursor down one row.
    if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
    lcd.at(row, col, curWord); //Print the word that was erased from the previous
row.
    col += curWord.length(); // Move cursor to past the end of the current word
segment.
    if (col > COLS - 1) { //Will be used for later in case the word exceed the
whole line.
      row += col / 20; //Number of rows the text exceeded.
      col = (col + 1) % 20; //The remaining column index after the text has
spilled over.
      if (row > ROWS - 1) {
        lcd.empty();
        row = 0;
        col = 0;
      }
    }
    curWord = "";
  }
}

//Display a Character
void disChar() {
  lcd.at(row, col, curChar); //Display a character at the current cursor position.
  curChar = "";
  col++; //Increase the cursor's column position.
}
```

Due

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;
```

```
void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  if (key) { //As long as key isn't zero (a null character), it will print.
    Serial.write(key);
    key = 0;
  }
}
```

Fixing Pre-emptive Wrap

Arduino Uno

- I simply adjusted the condtional statement for text wrapping:

From

```
if (col > COLS - 1) { //If word too long, wrap.
```

To

```
if (col > COLS) { //If word too long, wrap.
```

- It worked and the text can use the whole line.

Trying to Fix Long Text Wrap

- The issue is that when a word is longer than the screen's width, it writes over itself, erasing what was previously there.

Disfunctional

- I made an attempt to decifer long words verus not long.

```
void updateCursor() {
  if (col > COLS) { //If word too long, wrap.
    if (curWord.length() + spaceC + 1 > COLS) { //Word too long for whole line.
      curWord = "";
```

```
      row++;
      overflowRow();
      col = 0;
      spaceR = row;
      spaceC = -1;
      Serial.println(String(col) + "; " + String(row) + "; " + String (curWord) +
"; "); //Diagnostics
    } else {
      Serial.println(curWord.length());
      row = spaceR; //Start at the space index row.
      col = spaceC + 1; //Start just after space index column.
      for (int i = 0; i < curWord.length(); i++) { //Insert number of space
require to "erase" the word.
        spaces += " ";
      }
      lcd.at(row, col, spaces); //Erase the last word.
      spaces = "";
      col = 0; //Reset cursor at column 0.
      row++; //Move the cursor down one row.
      overflowRow();
      lcd.at(row, col, curWord); //Print the word that was erased from the
previous row.
      col += curWord.length(); // Move cursor to past the end of the current word
segment.
      curWord = "";
    }
  }
}
```

"overflowRow();"

```
void overflowRow() {
  if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
    }
}
```

- This ruined the main function, but the cursor was moved down to the next line, allowing for long text to not be erased but still display.

Semifunctional

```
void updateCursor() {
  if (col > COLS) { //If word too long, wrap.
    if (row - spaceR > 0) { //If the space row index was on the pervious line.
      curWord = "";
      row++;
      overflowRow();
```

```
      col = 0;
      spaceR = row;
      spaceC = -1;
    } else {
      row = spaceR; //Start at the space index row.
      col = spaceC + 1; //Start just after space index column.
      for (int i = 0; i < curWord.length(); i++) { //Insert number of space
require to "erase" the word.
        spaces += " ";
      }
      lcd.at(row, col, spaces); //Erase the last word.
      spaces = "";
      col = 0; //Reset cursor at column 0.
      row++; //Move the cursor down one row.
      overflowRow();
      lcd.at(row, col, curWord); //Print the word that was erased from the
previous row.
      col += curWord.length(); // Move cursor to past the end of the current word
segment.
      curWord = "";
    }
  }
}
```

- Almost worked. It displayed the normal words normally, and the long word was displayed without writing over itself. However, when two lines were exceed, it added an extra empty line in-between the two line, which is not what I want and is inefficient.

Half-functional

- Removed unnecassry space index setters (spaceR and spaceC).

```
void updateCursor() {
  if (col > COLS) { //If word too long, wrap.
    if (row - spaceR > 0) {  //A way of checking that the text has already been
moved down but has not had a space in it.
      curWord = "";
      row++;
      overflowRow();
      col = 0;
    } else {
      row = spaceR; //Start at the space index row.
      col = spaceC + 1; //Start just after space index column.
      for (int i = 0; i < curWord.length(); i++) { //Insert number of space
require to "erase" the word.
        spaces += " ";
      }
      lcd.at(row, col, spaces); //Erase the last word.
      spaces = "";
      col = 0; //Reset cursor at column 0.
      row++; //Move the cursor down one row.
```

```
      overflowRow();
      lcd.at(row, col, curWord); //Print the word that was erased from the
previous row.
      col += curWord.length(); // Move cursor to past the end of the current word
segment.
      curWord = "";
    }
  }
}
```

- Erased the first word on the next line, but did not add a whole empty line. Also, when starting at row = 0, the text jumped down a line.

Almost There

- Adjusted "overflowRow() {}" to set the space index row to a value that normally can't exist, but still helps with long text.

```
void overflowRow() { //Used to rest screen and row position if the row exceeds the
screen's height.
  if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
      spaceR = -1; //Setting it to a "null" value, in a way.
    }
}
```

- Almost worked. The text didn't re-display itself at all, but the first character in the next line was erased with the long word.

So, So Close

- Adjusted the column index (col) inside the long text adjustment to be set to one value after the next character is displayed.

```
    if (row - spaceR > 0) {  //A way of checking that the text has already been
moved down but has not had a space in it.
      curWord = "";
      row++;
      overflowRow();
      col = 1; //Start one value after the character is moved down a line.
    }
```

- Worked almost as intended. When moving onto a blank screen, there was an extra space at the beginning.

Functional

- Added a work-around for the character not being displayed on the next line when the word was long.

```
void overflowRow() { //Used to rest screen and row position if the row exceeds the
screen's height.
  if (row > ROWS - 1) { //If cursor exceeds rows, reset rows and screen.
      row = 0;
      lcd.empty();
      spaceR = -1; //Setting it to a "null" value, in a way.
      lcd.at(0, 0, curWord.substring(curWord.length() - 1)); // Display the last
character of the curWord. A normal-lengthed word overwrites the place value, but
this is used for long-lengthed words.
    }
}
```

Discovery

Due

- PallaxLCD library doesn't work with the Arduino DUe because of its processor.

ASCII Serial for Arduino Uno and Due

- I tried using the serial output to feed the LCD characters directly using ASCII, and it worked. Characters were displayed on the screen without needing the special library.
- I guess I just won't have the Arduino Due be in a special format, but I can still try for the Arduino Uno to display the characters in a nice and neat format.

Code for Basic Text Operation

Arduino Due

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}
```

```
void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  if (key) { //As long as key isn't zero (a null character), it will print.
    Serial.write(key);
    key = 0;
  }
}
```

Arduino Uno

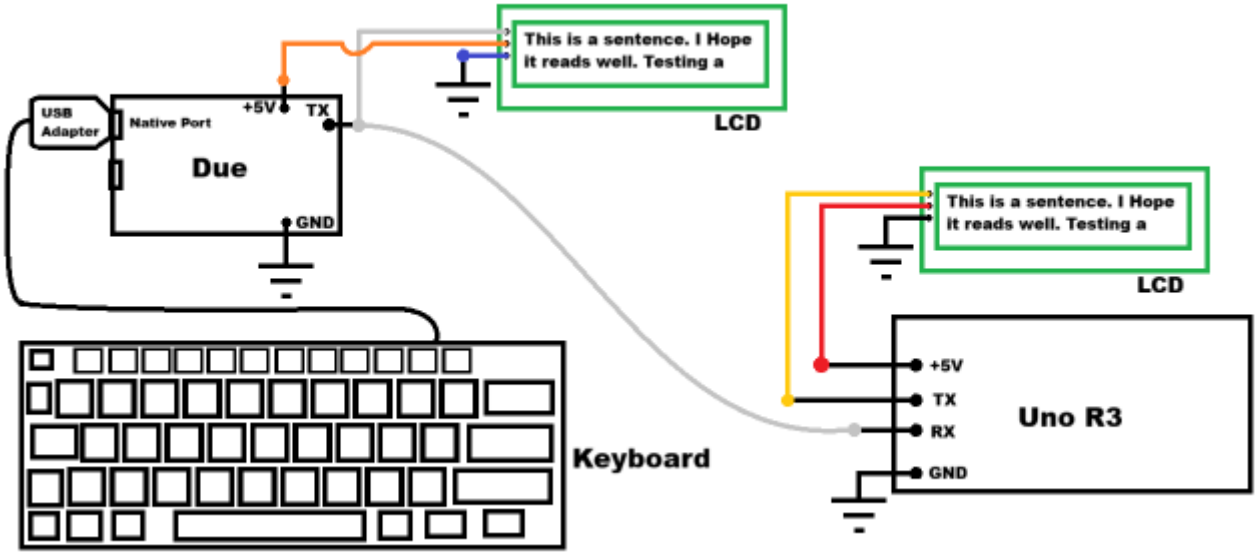
```
void setup () {
  Serial.begin(9600);
}

void loop () {

  if (Serial.available()) {
    Serial.write((char) Serial.read());
  }

}
```

Setup for Basic Text Operation



Implementing Special Characters

Experimentation

Morse Code Ideas

- I will use the enter key. It cannot be read by the Arduino Due as an ASCII character, but it can return an Oem key.

- LED on both receiving and sending end.
- Receiving end will not display the special character used for the enter key, but the Due's LCD will.
- One character will represent enter key pressed and turns the LED on, another character represents the key released and turns the LED off.

Detecting Oem Key

Code

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;
int OemKey = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  OemKey = keyboard.getOemKey(); //Assign the Oem key to OemKey.
  Serial.println("key pressed: " + String(key) + "; OemKey " + OemKey);
}

void keyReleased() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  OemKey = keyboard.getOemKey(); //Assign the Oem key to OemKey.
  Serial.println("key released: " + String(key) + "; OemKey " + OemKey);
}
```

- The keys only activated when pressed or released, as expected.
- The OemKey returned for the enter key had a value of 40.
- Other keys had different OemKey values.
- Looking at the Parallax LCD's datasheet, ASCII values between 233-247 do nothing for the LCD, so I could use these two of these for morse code operation.

Using null ASCII values for Morse On and Off

Due

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;
int OemKey = 0;
int LED = 2;

void setup(){
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  OemKey = keyboard.getOemKey(); //Assign the Oem key to OemKey.
  if (key != 0) { //If key isn't 0.
    Serial.write(key);
  }
  if (OemKey == 40) {
    digitalWrite(LED, HIGH);
    Serial.write(240);
  }
}

void keyReleased() {
  OemKey = keyboard.getOemKey(); //Assign the Oem key to OemKey.
  if (OemKey == 40) {
    digitalWrite(LED, LOW);
    Serial.write(241);
  }
}
```

Uno

```
int LED = 2;
int key = 0;

void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
```

```
}

void loop() {
  if (Serial.available()) {
    key = Serial.read();
    Serial.write(key);
    morseCode();
  }
}

void morseCode() {
  if (key == 240) { //If enter key was pressed.
    digitalWrite(LED, HIGH);
  } else if (key == 241) { //If enter key was released.
    digitalWrite(LED, LOW);
  }
}
```

- It worked. Characters were displayed as normal, but the enter key turned the light on and off.

Finding OemKey Values for Different Keys

- The code is meant for returning the OemKey only. I will be recording the corresponding values as I go along.

Due Code

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;
int OemKey = 0;

void setup(){
  Serial.begin(9600);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
  OemKey = keyboard.getOemKey(); //Assign the Oem key to OemKey.
  Serial.println("key pressed: " + String(key) + "; OemKey: " + String(OemKey));
}
```

Results

- Most values read a key of "0", except for the enters keys, operators (+,-,/,*), and any normal characters. All the special keys read a key value of zero, but had a non-zero value for the OemKey.
- Results are as follows:

Arrow Keys

- *Up:* **82**
- *Down:* **81**
- *Left:* **80**
- *Right:* **79**

Cluster (Home, End, etc.) Above Arrow Keys

- *Insert:* **73**
- *Home:* **74**
- *Page Up:* **75**
- *Page Down:* **78**
- *End:* **77**
- *Delete:* **76**

F-Keys

- *F-1:* **58**
- *F-2:* **59**
- *F-3:* **60**
- *F-4:* **61**
- *F-5:* **62**
- *F-6:* **63**
- *F-7:* **64**
- *F-8:* **65**
- *F-9:* **66**
- *F-10:* **67**
- *F-11:* **68**
- *F-12:* **69**

Numpad

- *0:* **98**
- *1:* **89**
- *2:* **90**
- *3:* **91**
- *4:* **92**
- *5:* **93**
- *6:* **94**

- *7:* **95**
- *8:* **96**
- *9:* **97**
- *Del:* **99**
- *Enter:* **88**

Miscellaneous

- *Backspace:* **42**
- *Enter (Main Key):* **40**
- *Weird Page-Like Symbol next to Right Ctrl Key:* **101**
- *Tab:* **43**
- *Esc:* **41**
- *PrintScreen:* **70**
- *ScrollLock:* **71**
- *PauseBreak:* **72**

Functional Version

Due

```
#include <KeyboardController.h>

// Initialize USB Controller
USBHost usb;

// Attach Keyboard controller to USB
KeyboardController keyboard(usb);

int key = 0;
int OemKey = 0;
int LED = 2;
const int morseKey = 77;

boolean screenPower = true;

void setup(){
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
  delay(10);
  Serial.write(12); //Clear screen.
  delay(10);
}

void loop(){
  usb.Task();
}

void keyPressed() {
  key = keyboard.getKey(); //Assign the key pressed to "key".
```

```
  OemKey = keyboard.getOemKey(); //Assign the Oem key to OemKey.
  if (key == 0) {
    sendCommand(OemKey);
  } else {
    Serial.write(key);
    if (OemKey == 40 || OemKey == 88) { //Enter Keys
      Serial.write(13); //Return to next line.
    }
  }
}

void keyReleased() {
  OemKey = keyboard.getOemKey();
  if (OemKey == morseKey) {
    Serial.write(15);
    digitalWrite(LED, LOW);
  }
}


void sendCommand(int code) {
  switch (code) {

    case 0:
      break;

    //Morse Code Key
    case morseKey:
      Serial.write(14);
      digitalWrite(LED, HIGH);
      break;

    //Arrow Keys
    case 81: //Down
      Serial.write(10); //Cursor down.
      break;
    case 80: //Left
      Serial.write(8); //Cursor left.
      break;
    case 79: //Right
      Serial.write(9); //Cursor right.
      break;

    //Cluster
    case 76: //Delete
      //Just replace with space.
      Serial.write(" ");
      Serial.write(8);
      break;
    case 74: //Home
      Serial.write(12); //Refresh Display ("Form Feed")
      break;

    //Misc
```

```
    case 42: //Backspace
      //Functions like backspace.
      Serial.write(8);
      Serial.write(" ");
      Serial.write(8);
      break;
    case 43: //Tab
      Serial.print("   "); //Three spaces.
      break;
    case 41: //Escape
      powerLCD();
      break;

    //Default
    default:
      break;
  }
}

//Turns LCD on or off.
void powerLCD() {
  if (screenPower) {
    Serial.write(21); //Turns LCD screen off. Doesn't erase characters.
    screenPower = false;
  } else {
    Serial.write(24); //Turns LCD screen on.
    screenPower = true;
  }
}
```

- The switch-case is used to detect different OemKey codes to do different functions with the LCD screen, including backspace, cursor movement, screen display, clearing the screen, and morse code LED.
- The reason the enter keys are not in the switch-case is because they have non-zero key values. Also, the value is not interpretted by the LCD as anything, so I had to use the Oem code. The way I set it up works properly.

Uno

```
int LED = 2;
int key = 0;

void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);

}

void loop() {
  if (Serial.available()) {
    key = Serial.read();
    Serial.write(key);
```

```
    morseCode();
  }
}

void morseCode() {
  if (key == 14) { //If End key was pressed.
    digitalWrite(LED, HIGH);
  } else if (key == 15) { //If End key was released.
    digitalWrite(LED, LOW);
  }
}
```

- Worked as intended, turning on the LED as necessary and receiving the ASCII values from the Arduino Due.

Miscellaneous Documentation

Documentation Overview

Laser Safety

Hazards:

Class FDA	Class IEC	Laser Product Hazard	Product Example
I	1, 1M	Considered non-hazardous. Hazard increases if viewed with optical aids, including magnifiers, binoculars, or telescopes.	• laser printers • CD players • DVD players
IIa, II	2, 2M	Hazard increases when viewed directly for long periods of time. Hazard increases if viewed with optical aids.	• bar code scanners
IIa, II	2, 2M	Hazard increases when viewed directly for long periods of time. Hazard increases if viewed with optical aids.	• bar code scanners
IIIa	3R	Depending on power and beam area, can be momentarily hazardous when directly viewed or when staring directly at the beam with an unaided eye. Risk of injury increases when viewed with optical aids.	• laser pointers
IIIb	3B	Immediate skin hazard from direct beam and immediate eye hazard when viewed directly.	• laser light show projectors • industrial lasers • research lasers
IV	4	Immediate skin hazard and eye hazard from exposure to either the direct or reflected beam; may also present a fire hazard.	• laser light show projectors • industrial lasers • research lasers • medical device lasers for eye surgery or skin treatments

Table made by U.S. FDA

- My laser is a class IIIa laser rated at 5 mW.
- Hazards inlcuding staring into or directly at the beam. The hazard of the laser increases when using optics to stare at the beam.
- Classes II through IV must include a warning symbol depicting class and power output.

Effects

- Primary hazardous effects of lasers are to the skin and eyes.
- Lasers can cause heat and photochemical damage.
- Excessive Visible light causes damage to the retina.

Other Info

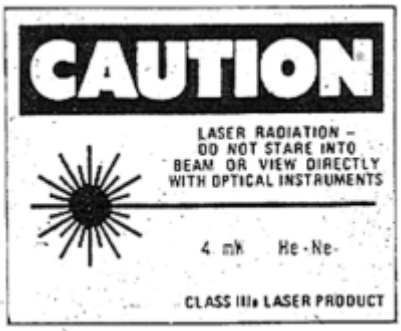
- The Laser beam can reflect off surfaces, and the divergence of the beam can be changed.
    - Unchanged divergence with a flat surface (reflective hazard).
    - Increased diveregence angle (more spread out) with a convex surface.
    - Focused point, before diverging out with a concave surface.

Safety Practices

- If seeing a laser pointing into your eye, look away from the source, and exit beam path.
- Keep beam path away from eye level.
- Terminate or diffuse laser light after hitting target.
- Only operate the laser when necessary.
- Keep unecessary objects out of path of lasers.
- Do not move unecessary optics into the path of the beam.
- Protective eyewear and clothing for high-powered lasers.
- Keep bodies parts out of the beam path.

Examples

Class IIIa Laser Warning Label



Safety Practices I Will Take

- The laser will be mounted far above eye level, to avoid possible accidental eye exposure.
- Since the photodiode has a convex surface, I will align the laser to touch the middle or just to the side away from the hallway, to allow for safe beam divergence.
- I consider using a small metal enclosure/casing around the photodiode, with an open end for the laser to enter through.
- When aligning the laser beam, contact of any body part in the path of the laser will be avoided.
- The laser shall not be stared at.
- Alignment of the laser will be done by using a paper background behind the photodiode, and aligned from behind or on the side of the laser.

- When working on mounting the photodiode or any coverings, the laser will be disconnected from power to avoid accidental powering and possible exposure.
- The warning label shall not be removed from the laser.
- Possible laser warning label at the termination of the laser path.

Notes

- For the final construction there are some things I didn't do. This includes mounting an enclosure around the photodiode. The photodiode mount was high enough and the laser weak enough that any reflection off the rail wasn't going to result in strain to the user's eyes. Also, I did use my hand for aligning the laser, by seeing where it went, but the laser was far too weak to cause bodily harm. In my laser safety "practices" section, I was taking an over-extreme precaution that wasn't necessary for my class of laser.