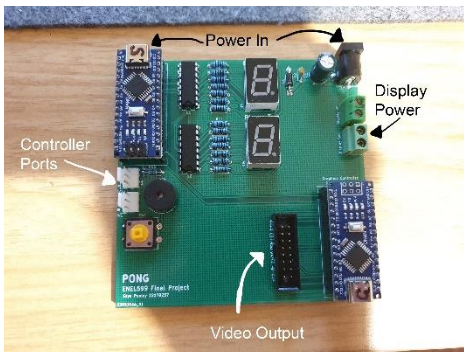
ENEL599 Final Project

Premise
I decided to make a pong gaming system using an LED display panel. The Arduino will output its display to a 32x16 p10-8s RGB LED panel. Player input can be gathered using two potentiometers which map their turn range to the paddle position for that player. Score should be output to two 7-segment displays. A speaker or buzzer should beep when the ball hits a panel or goes out of bounds.

Inspiration
https://www.youtube.com/watch?v=cm83RIhDbwo

Helpful Libraries and Guides
https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/new-wiring
https://www.circuitbasics.com/programming-with-classes-and-objects-on-the-arduino/

Documentation


Connect the video output from the PCB to the video input of the matrix panel. Connect the screw terminal power outputs to the power input of the matrix panel. Then connect the two controllers to the ports on the left side. Then connect power to turn on the system.

Use the dials on the controllers to position each player's "paddle", then the player who's turn it is to serve can press the white button on the controller to serve the ball. If a player misses the ball when it comes their way then the opposing player will be given a point. The first player to reach three points (adjustable in software) will win the game. After this, press the yellow button to play again.

Parts list
- RGB LED Panel P10-8s
- Arduino nano for panel driving
- Second Arduino for game simulation, player IO.
- 2x 7-segment display
- Shift register 74hc595
- 5V 15W power supply.
- 2x 10kr Potentiometers with knobs

Changelog
Panel Driver Prototyping
After attempting to interact with the shift-registers in the display panel I decided that I would not have enough time to write my own driver code at this stage. I decided to use the Adafruit LEDMatrix library to drive the panel.
My initial prototype used a breadboard to connect the Arduino to the LED panel but encountering some instability issues I decided to move this section to a soldered prototyping board to eliminate inconsistent connections.

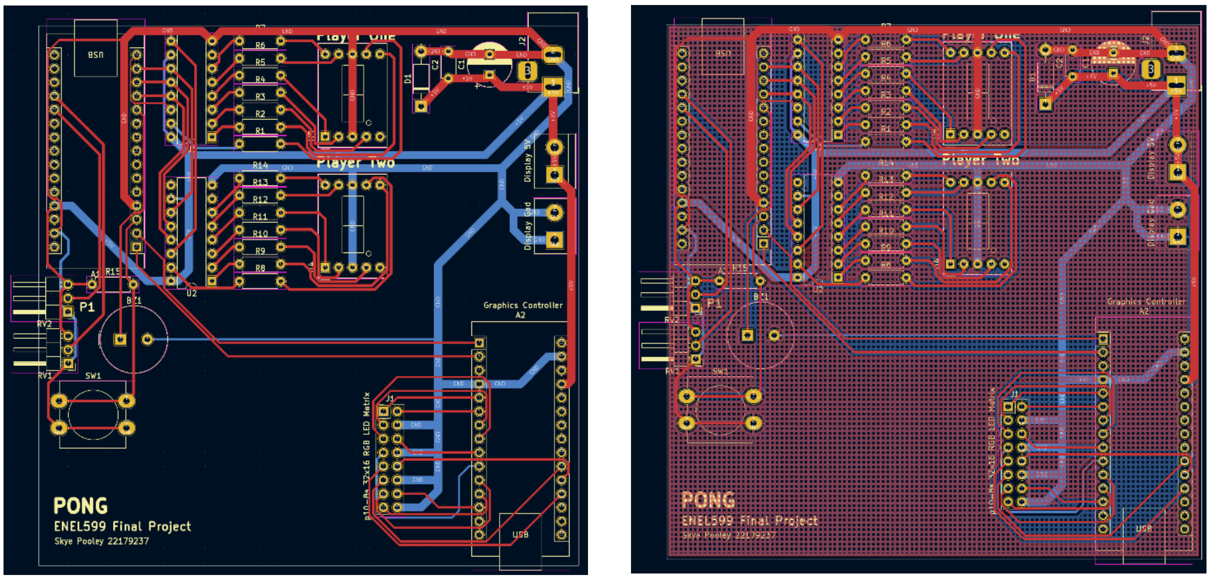
I ran a test to check if the line-select wires were incorrectly connected by lighting up each row of the display in order. I noticed some lines lighting up in the incorrect order so I swapped the line select pins in software, this resolves my image distortion.

7-Segent Display Prototyping

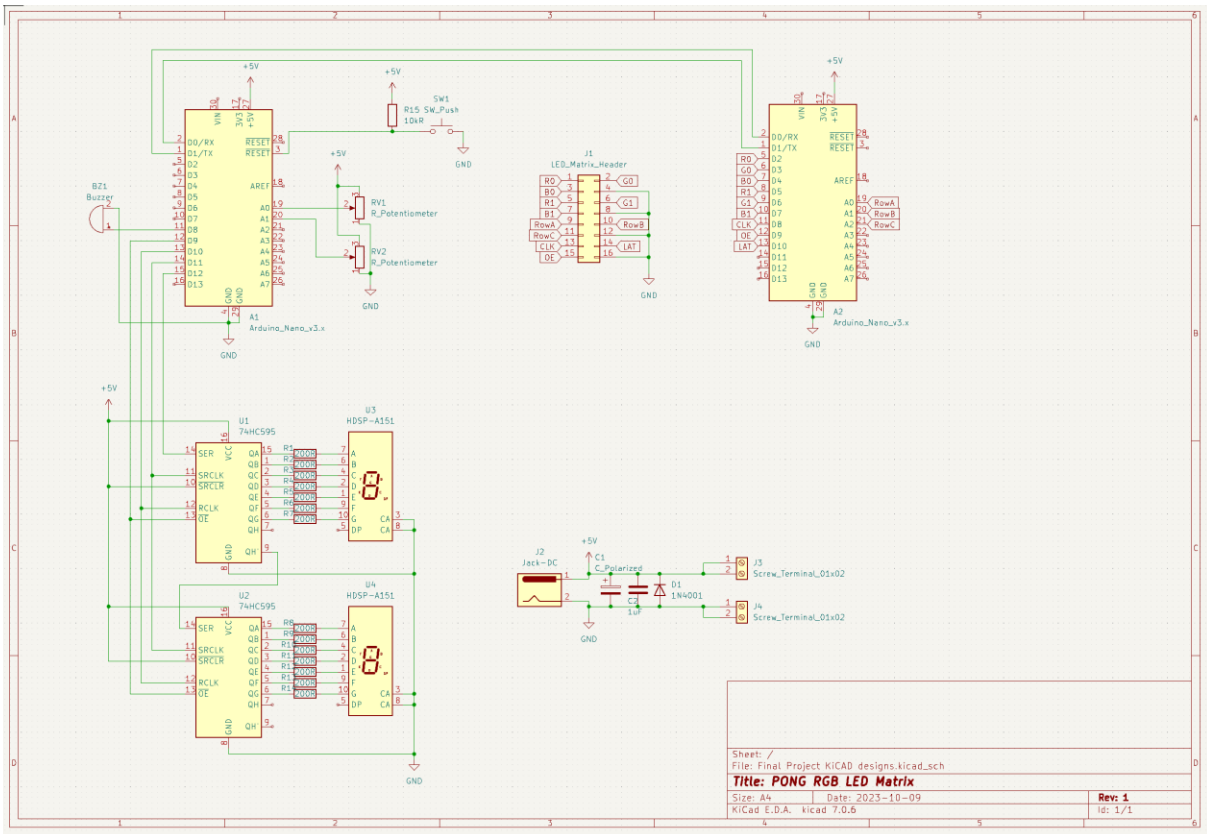
To display the scores, I wanted to use 7-segment displays driven by shift registers to minimize pin use on my Arduino. Using a second Arduino I wired up two displays, one for each shift register and chained the registers together.
After correcting some misplaced wires, I was able to display numbers on the displays by mapping segment on/off states to bits in a byte and shifting this byte out into the registers.

PCB Design

To integrate my two prototypes into a cohesive system, I decided to design and order a PCB. I designed a schematic in KiCAD and then imported the nets into the KiCAD PCB designer. After making all connections and ensuring the design passed DRC I sent the files to JLCPCB for manufacturing.



PCB Design shown with and without ground fills visible.



Schematic.

Serial Communication

The computation for the game needs to be split between two Arduinos due to memory constraints. The PCB design connects the two Arduinos via serial so that one can handle video rendering while it receives information on entity positioning via serial from the other Arduino which handles IO and game simulation.

Numbers to transmit:

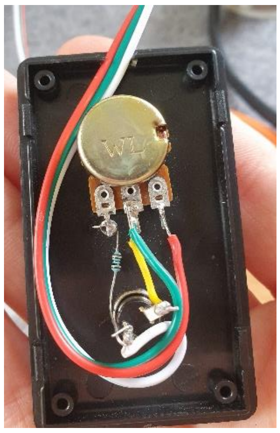
1. Paddle A position – 0-15 (4 bits)
2. Paddle B position – 0-15
3. Ball horizontal pos – 0-31 (5 bits)
4. Ball vertical pos = 0-15

We will need 5 bits to transmit the value of each number leaving three bits for the flag. Three flag bits gives us 8 possible flags, more than currently needed.

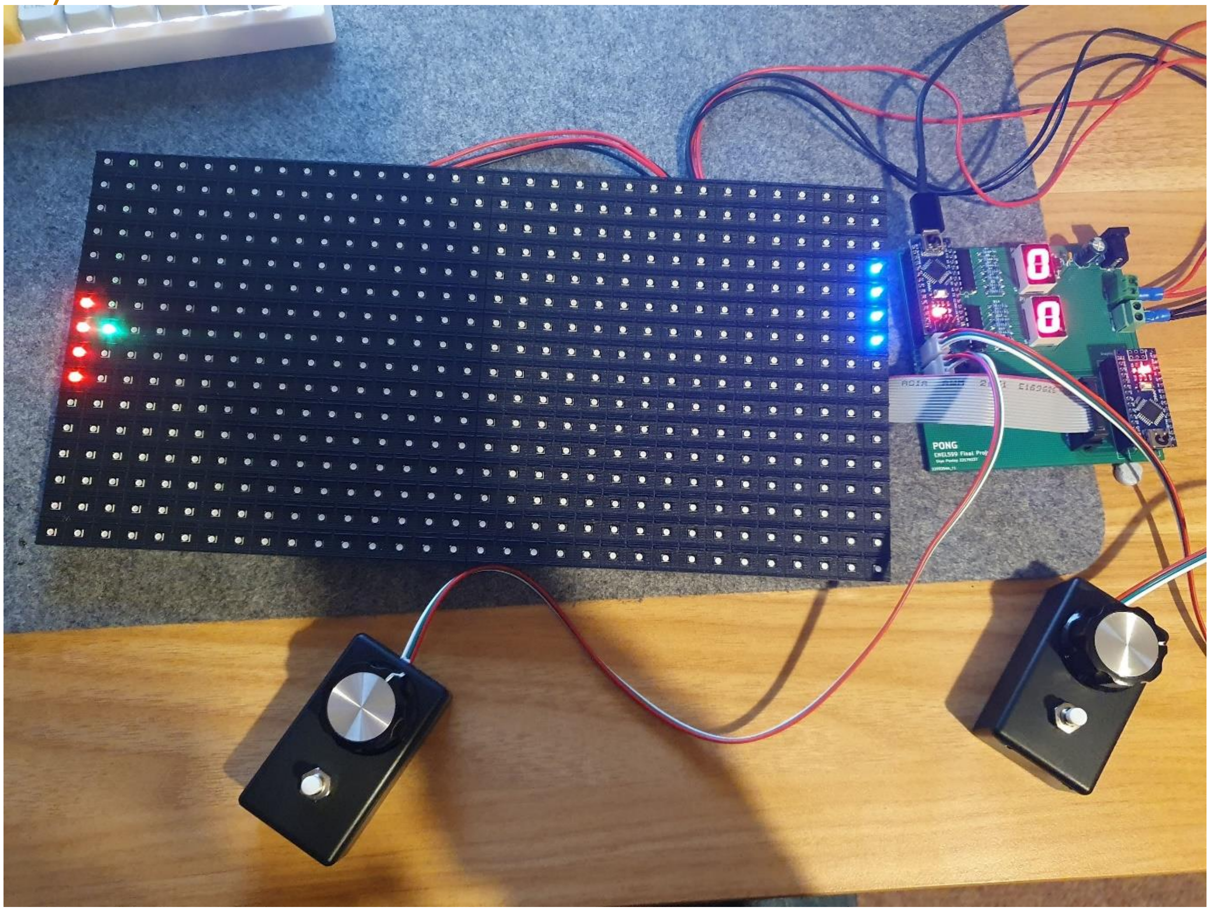
Because this totals 8 bits we can transmit each value in a single byte rather than transmitting flags and values separately. This will simplify the serial protocol as we don't need to worry about matching up the bytes.
Each transmitted byte is in the form 0bFFFVVVVV where F is the flag and V is the value.

Controllers

The player input needed a dial for paddle positioning and a button to serve the ball on a player's turn. The cables that I had on hand were only three pin so I needed a way to transmit both the potentiometer and button input over one wire. To do this I wired the potentiometer ground through a 1k resistor to limit its output range to 0.5-5V. The button would then pull the output down to 0V.



Fully Functional Game



I used the lessons that I learned from the prototypes to write the code for the first fully integrated version.