LED Chess Board

School of Engineering and Applied Sciences
Harvard University

Cambridge MA

Description: Interactive Chessboard with Legal Moves

By: Evangelos Kassos and Gianluca Trofa

Class: Engineering Sciences 50
Instructor: Marko Loncar & Christopher Lombardo

May 2022

ABSTRACT

Learning how to play chess is complicated and challenging for many newcomers. Oftentimes, players who recently started practicing find it hard to remember the legal moves for each piece type. We set out to solve this problem by creating an interactive chessboard that registers the piece which the player picks up and displays possible movements. Our original design incorporated a total of 64 circuits, one installed under each tile. Each circuit was cut using a header male/female header design. The male header end was placed in the middle of each tile, so that its two pins would protrude the tile. We incorporated the female header in the chess pieces. Therefore, when the piece was placed on a tile, the circuit was completed, sending a high signal. The 8 signals from the tiles of each row were collected by a shift register and sent to the Arduino serial monitor for monitoring. Our Arduino code read the array of 64 signals by the serial monitor and tracked when there were changes in the string to detect which piece was picked up. Using dedicated functions for each piece type, our code generates the possible moves for each piece and uses the FastLED Arduino library to light up the corresponding LEDs placed under the legal move tiles. When the piece is placed on a tile again, the code refreshes the board so that the next move is possible. Unfortunately, our shift registers proved to be unreliable, and we pivoted to a second design concept. We wrote code to generate a text-based chessboard that showed up on the serial monitor and allowed the user to register their moves. The chessboard representation refreshes and displays the possible moves, while the corresponding LEDs placed under the legal move tiles lit up. Once the tile that the user wants to place the piece on is registered, the board refreshes once more so that the next move is possible.

INTRODUCTION

"Every chess master was once a beginner."

– Irving Chernev

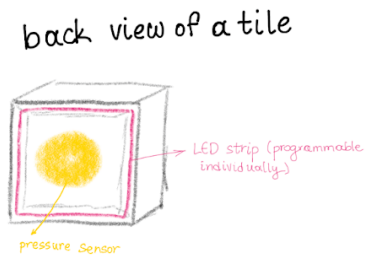
The premise of chess is simple: a board of 64 tiles, arranged in an eight-by-eight grid, contains 32 pieces, 16 for each player. Players take turns moving one piece at a time in an effort to checkmate the opponent's king. As chess enthusiasts, we set out to make the experience for chess players learning how to play smoother. We conducted user interviews with people who were interested in learning how to play chess, and experienced players. In all our interviews, one trend emerged: most players found it extremely difficult to remember the legal moves for each piece throughout the first few games they played. Having to stop the game to reference moves from print or online guides was a major friction point in the learning process and compelled many to stop learning very early on in the process (i.e., the first couple of games).

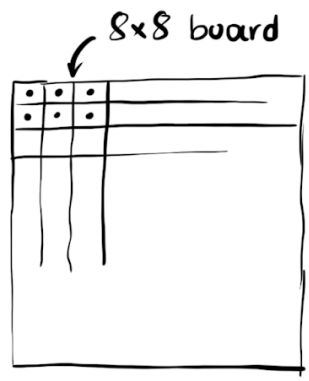
Given our data, we decided to focus on solving this issue. Therefore, we chose to create a chessboard design that interactively displays possible moves for each piece that a player picks up while being:

1. Easy to use
2. Efficient to use
3. Accessible
4. Approachable, and
5. Supports correct usage, making it easy to correct mistakes

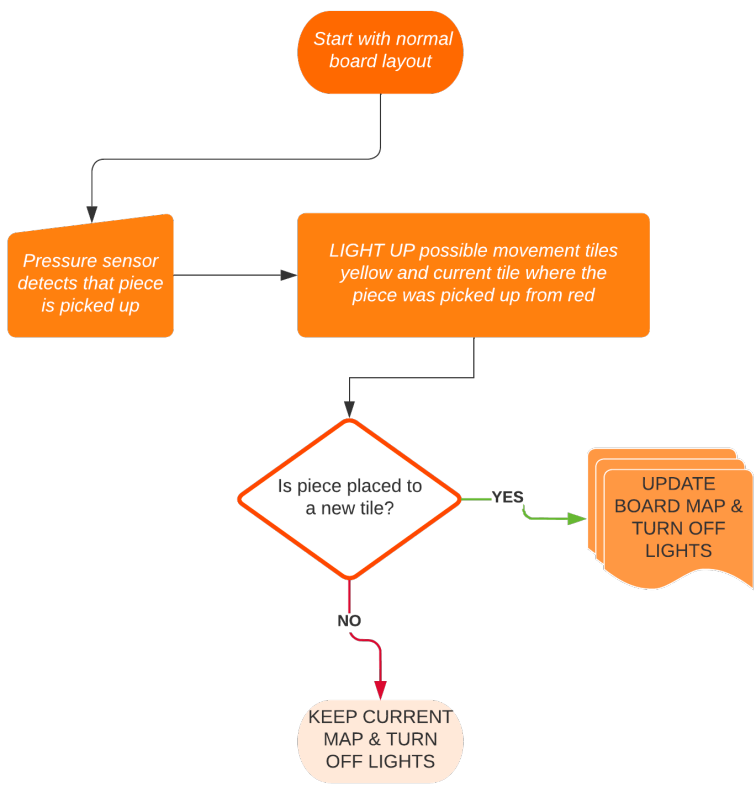
DESIGN

We began our design process by drawing inspiration from an LED Chessboard we saw online that lit spaces up pieces were placed on. We knew that instead of doing this, we would create a board that would go a step further and show where the actual piece would go.





We created these preliminary sketches of what our board layout would be and how each space on the board would look. We planned to line the LED strips along the walls of each space with the piece that dictated if a chess piece was on a space being cut out of the center of that piece.



As seen above, we then created a block diagram indicating the process for a move to occur in the game. This helped us in creating the actual board as we had to make sure each component functioned in an efficient way throughout the process to get our end result. We knew that there were many ways we could construct our board, and by giving ourselves room to interchange and adjust what components were being used to construct our board and complete each process on our block diagram, we could ultimately have success in completing our objective and creating a functional smart LED chess board.

PARTS LIST

Part description	Dimensions	Quantity
Individual-addressable LED light strip (model #: WS2812B)	15ft	3
Shift registers (model #: SN74HC165)		8
10KΩ Resistors		64
Breadboards		5
¼" clear acrylic	24" x 24"	1
⅛" black acrylic	24" x 12"	2
⅛" plywood	24" x 6"	4
Clear resin chess pieces		16
Blue resin chess pieces		16
Header switch male part		64
Header switch female part		32

PROJECT IMPLEMENTATION

****Note all figures can be found at end of document in appendix**

Step 1: Building the Chessboard

We started by 3D printing each tile of the board individually. At this time, we realized that it would be more time efficient to laser cut acrylic into the board shape. We took a 24" x 24" piece of ¼" clear acrylic and engraved spots to make a board (Figure A). Every other spot was glazed to create the effect of the two-colored

board tiles. We chose not to color the tiles and only use glazing so that the LEDs underneath each tile would show properly when lit up, without having discoloration because of the tile color. We also laser cut holes for buttons for which pieces would be placed on. As a reminder, these buttons would send information to our Arduino on which piece was getting lifted or placed on the board.

Step 2: Selecting the buttons

Our initial design idea entailed using pressure sensors to detect when pieces were being picked up from tiles. However, those parts were very expensive for 64 tiles, so we decided to use button switches instead. Unfortunately, after we made our chessboard design, we realized that the chess pieces we created (see step 4 below) were very light, and the button switch would not get activated when the button was placed on it. Therefore, we had to look for an alternate design choice. At first, we wanted to use reed switches, but those would arrive just 2 days before the design fair, which would have given us a very short timeframe to implement the wiring of our design.

Thus, we switched to the header as a button design. The female end of the header design would be glued into each chess piece, while the male end would go into the chessboard. Placing the piece on the header would complete the circuit, sending a high signal to the Arduino, and removing the piece would indicate a low signal at that spot. (Figure G)

Step 3: Walls

The walls were designed in a unique way to make sure there was no light leakage between tiles from the LEDs. The walls were also designed so that the LEDs could be easily interconnected and provide enough room for the LEDs and wiring to fit through (Figures B and C). We laser cut 14 wall pieces we designed with slits for the LED to be weaved through. LED strips were weaved through, and we attached them to 3 walls in each tile to make sure there was enough light for each tile. (Figure D)

Step 4: Pieces

We 3D printed our chess pieces with resin. Each print could fit roughly 6 pieces and took 10 hours to complete. (Figure E) We used a design for the chess pieces that we found on thingiverse (see references). Combined with reprints due to printer failure, it took about one and a half weeks to print all 32 chess pieces. Afterwards, all chess pieces were sanded, and we tapped a hole at the bottom of each piece with a ¼" drill bit in order to fit female header piece.

Step 5: Assembly

We started by creating and gluing small acrylic squares in each tile meant to make the button holes smaller, as we originally cut the holes for the bigger buttons and the headers were quite smaller. Then, we glued the female header pieces into the chess pieces, and tested to make sure that they were all working properly by creating a simple circuit with an LED and a male header part that the female header part at the bottom of the chess pieces would successfully complete. Afterwards, we soldered each male header so that pin A and pin B wires would be long enough to reach the breadboards with the shift registers that were sending information back to Arduino. We then glued each individual header into the board. We pulled each header's pin A and pin B wires through the walls of the board and attached them to the breadboards. (Figure H) Note that, as shown in the circuit illustrations, all pin A wires were connected to V_{CC}. Pin B wires were connected to the shift register pins, as shown in the circuit illustrations. Finally, we put the board down and attached the outside wooden walls. (Figure I)

Step 6: Coding

For the Arduino code, we used the FastLED library to control the individual-addressable LED light strip. We define a numerical value for each chess piece so that we can track their movement along the board. We then define the piece starting positions and initialize the board in its normal setup. We also define a 64 by 11 matrix that lists all LEDs that correspond to each tile. You can see our numbering conventions in the appendix. During setup, we initiate the LED strip and turn every LED to black (i.e., turn off) to make sure that the board is properly initialized from previous plays. Using our own *printBoard*() function, we print a

text-based representation of our board. Afterwards, the serial monitor prompts the user to input what piece they would like to pick up. Once the user enters a valid tile, we display the possible moves of the piece that was picked up using our own *movesBoard*() function and dedicated piece move generator functions. Legal moves are shown in white, while legal moves where a piece can be attacked are shown in blue. Once the player inputs a valid move (including placing the tile back to its picked-up tile), we use our own *refreshBoard*() function to update the board 8 by 8 matrix with the new positions and display the updated text representation. The game continues as players register moves and move pieces on the board. Finally, we track the pieces that have been captured by opponents to enable the promotion of pawns that reach the end of the board.

Step 7: Troubleshooting

After completing the wiring and using the code in the *register_test* Arduino code file, we observed that the shift registers were not reliable in producing accurate and reproduceable measurements, even though we tested all header wires and connections, and there were no floating pins. As we were not able to implement the header input as intended, we implemented a last-minute change. To ensure we had a working product, we had users input their move into our Arduino code. The players typed the position of the piece they wanted to pick up, the board lit up, the user typed the position where they wanted to place their piece, and then they were able to place the piece. Undoubtedly, this implementation was not as user-friendly or time efficient as the original design, but it achieved our design goal of a smart chessboard, which can show possible moves. In the end, our project was still very useful for new players, as it can help visualize where pieces can go.

TEAM MANAGEMENT

Gianluca was in charge of Mechanical Engineering tasks (i.e. laser cutting, design, soldering, drilling, etc.). Evangelos was in charge of Electrical Engineering (i.e. schematics, wiring, design, etc.) and coding. We both helped each other in each other's expertise areas; we made sure to have the one with better understanding and expertise lead in their area.

OUTLOOK AND POSSIBLE IMPROVEMENTS

If our budget was bigger, we could have ordered parts that would have made things simpler. More time would have allowed for more testing and construction of the final design (i.e., using magnetic reed switches instead of headers). Finally, placing the walls first and then gluing in the headers would have saved time.

ACKNOWLEDGEMENTS

Big thanks to Elaine, Steve, Charlie in the Makerspace for their help and advice, as well as Leo for the material inspiration.

ACKNOWLEDGEMENTS

Okay to share report, and photos and videos of the product, please do not share the chessboard code.

REFERENCES

Pieces:
https://www.thingiverse.com/thing:470700
Inspiration:
https://www.youtube.com/watch?v=SpNvPSR5WSE&ab_channel=CraftStatio

APPENDIX

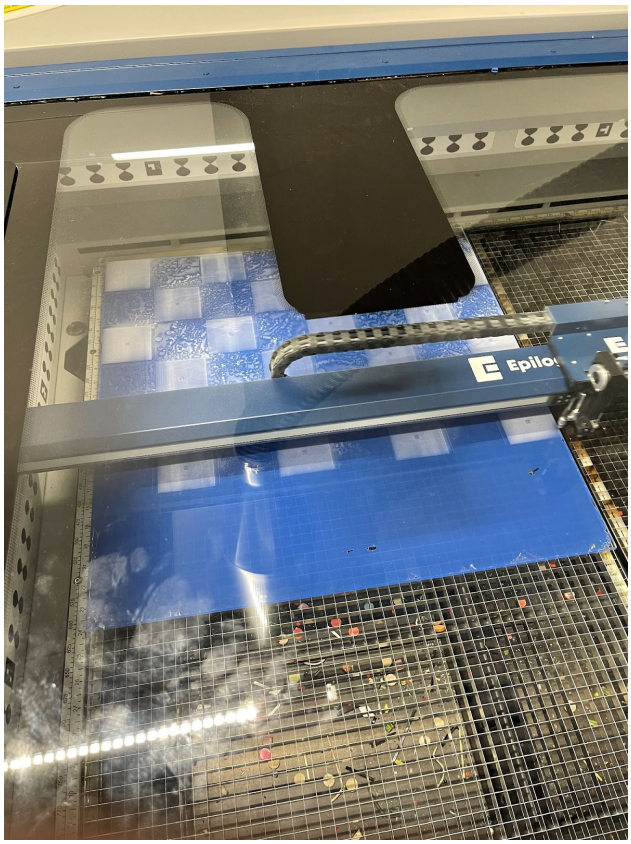


Figure A: Board being laser cut

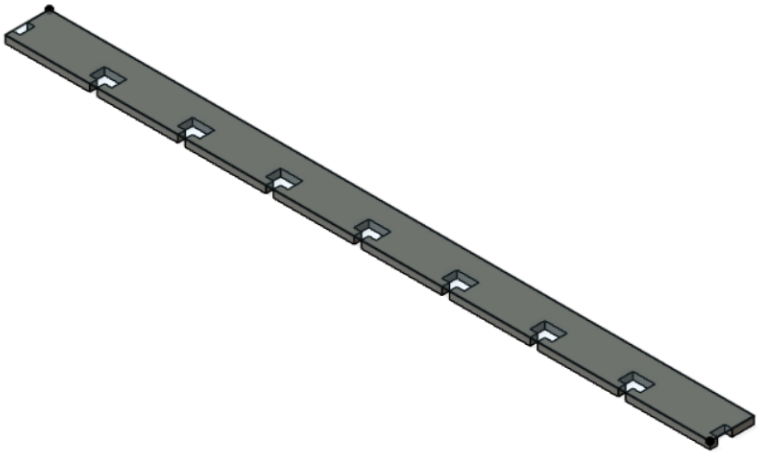


Figure B: Fusion 360 design of chess board wall

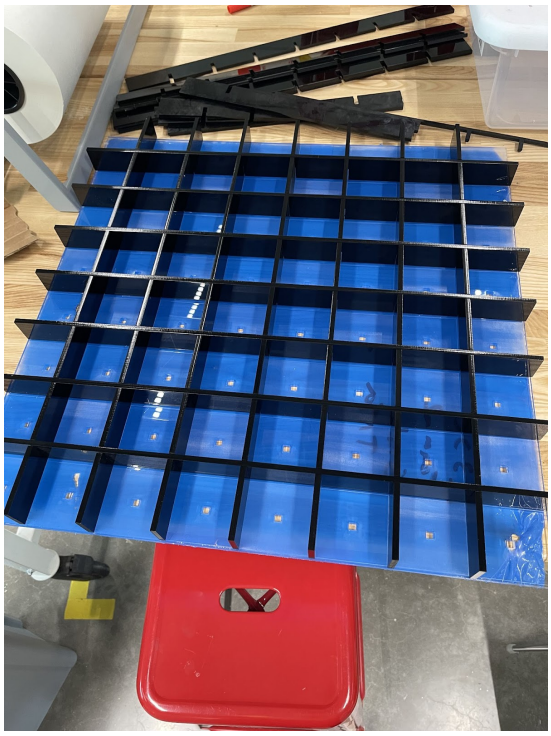


Figure C: Walls interlock design against the board

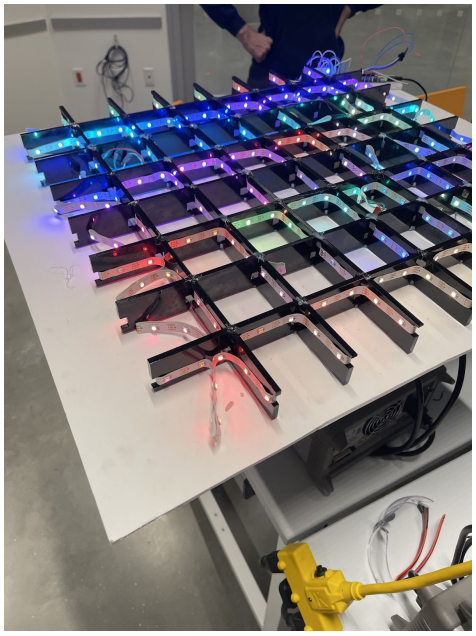


Figure D: Walls after all LEDs were weaved through and attached

Pieces:

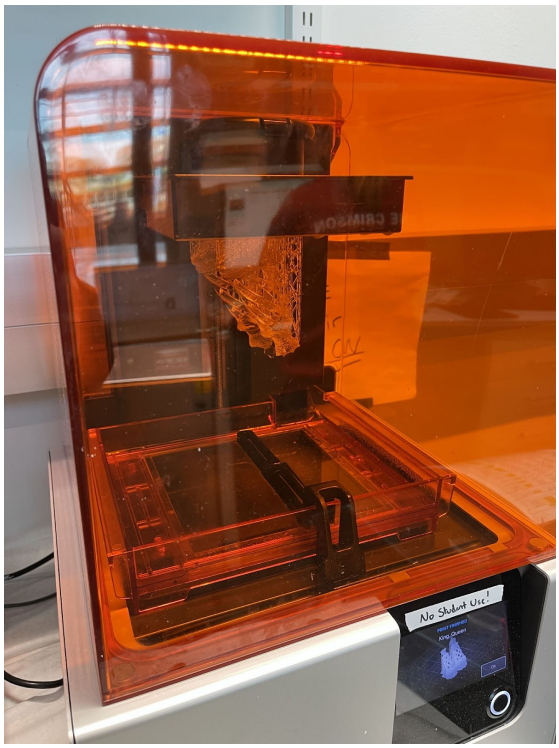


Figure E: 3D Resin Printing of King and Queen of clear pieces

Headers (what pieces were placed on):

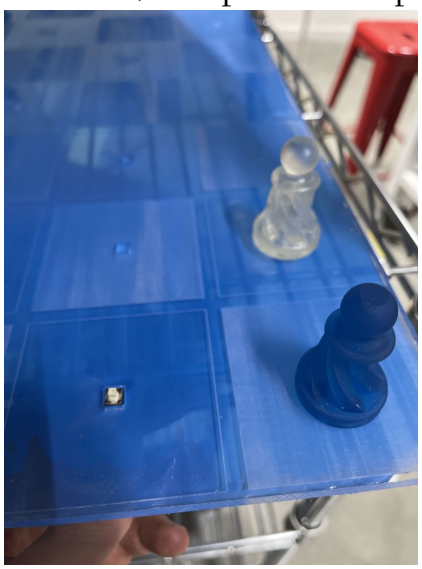


Figure F: Initial idea where piece would be placed on button through board

Figure G: Headers and how they could be implemented

Wiring (see end for circuit design):

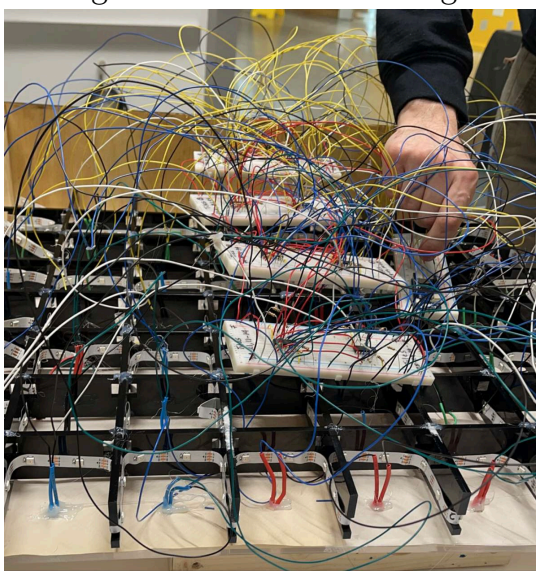


Figure H: Initial wiring when using multiplexes

Final Design:

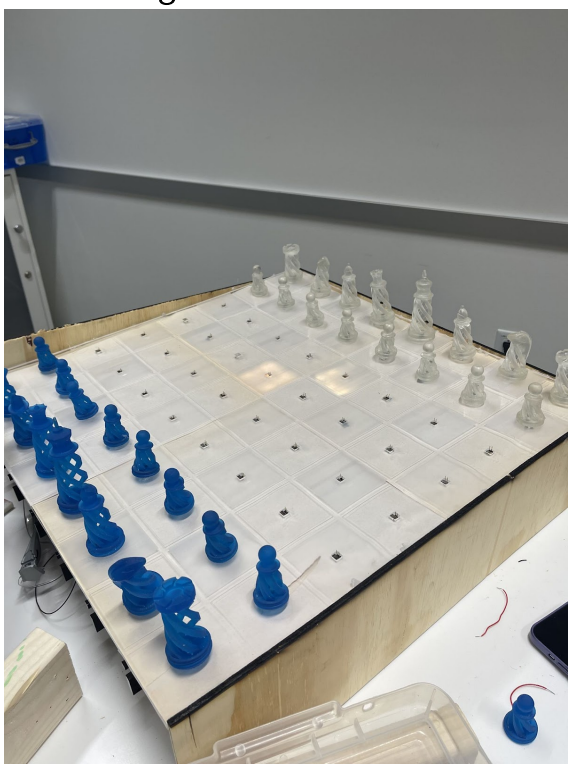
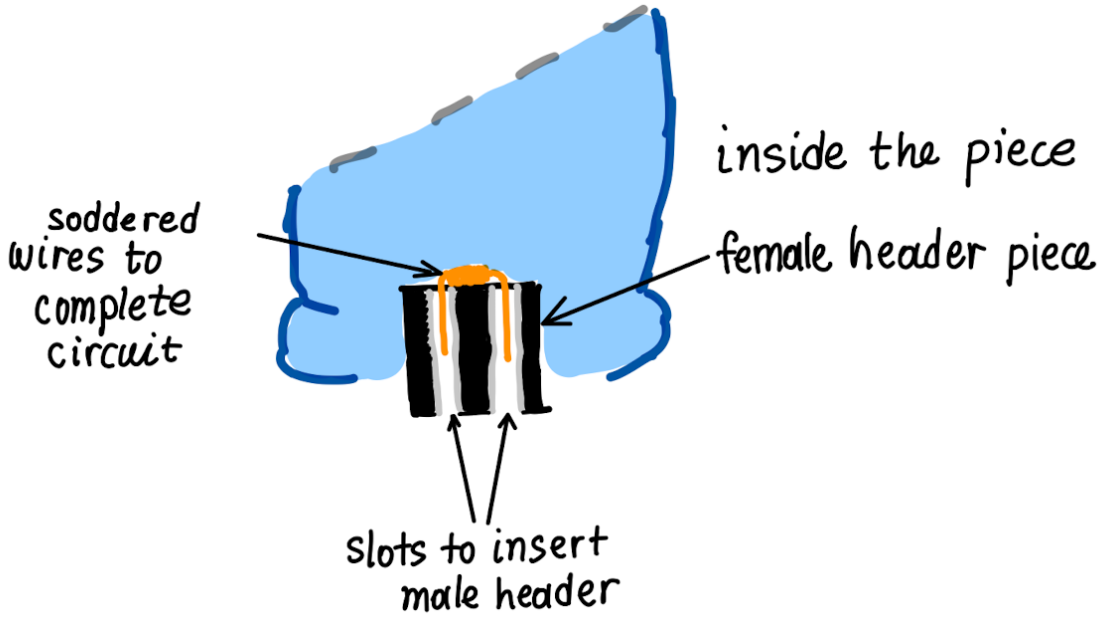
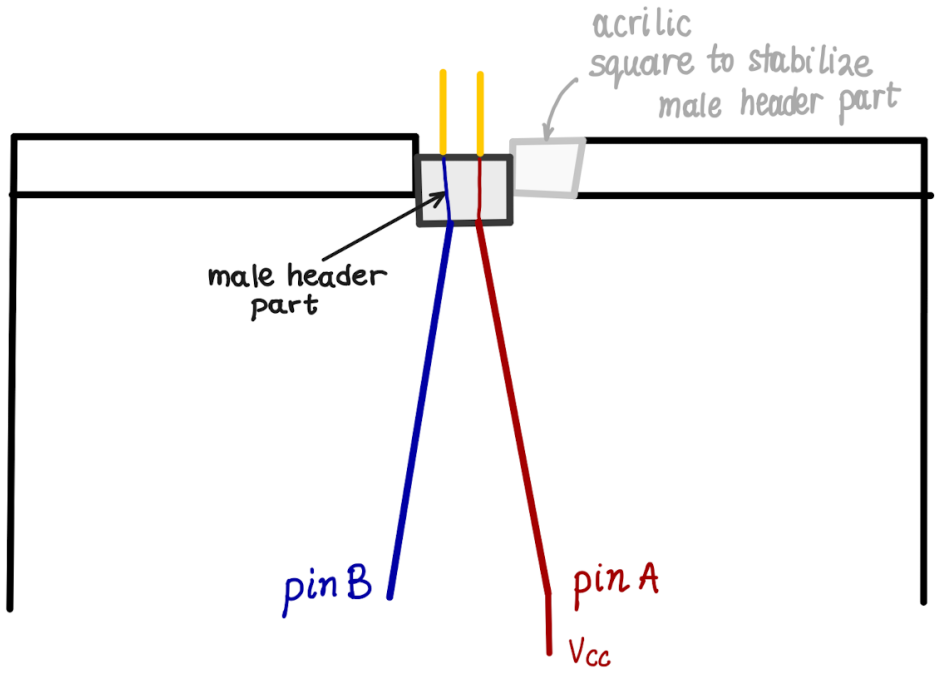


Figure I: Final product of board assembled with lit up spaces.

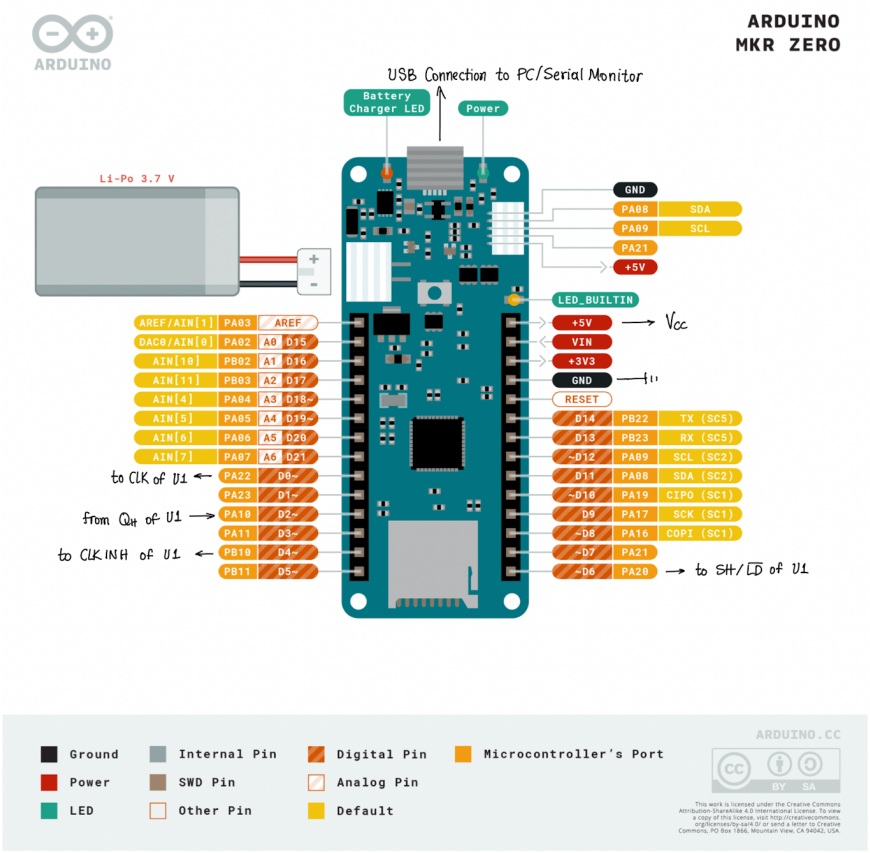
Piece design:

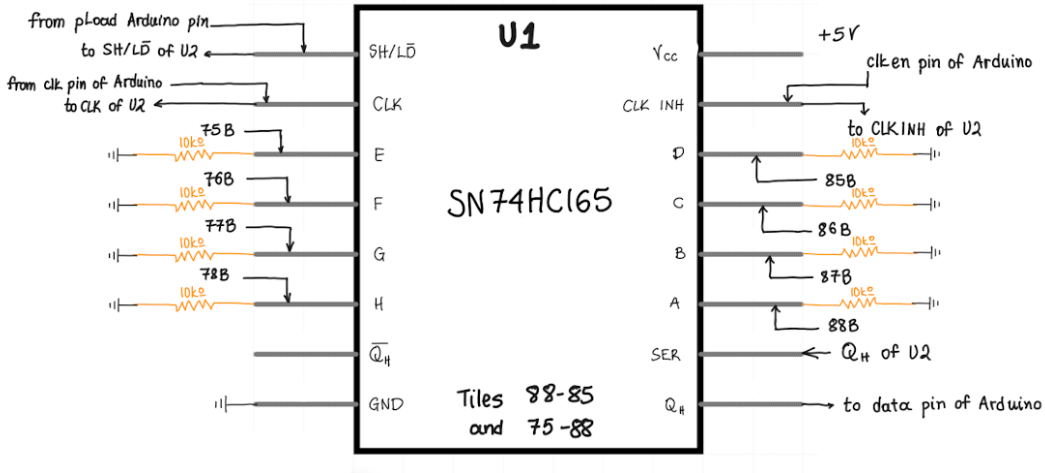


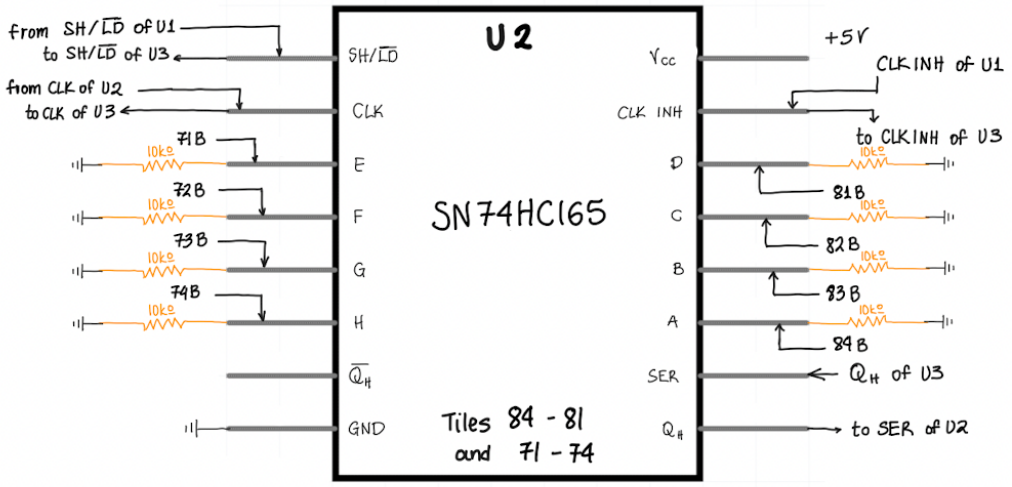
Tile design:

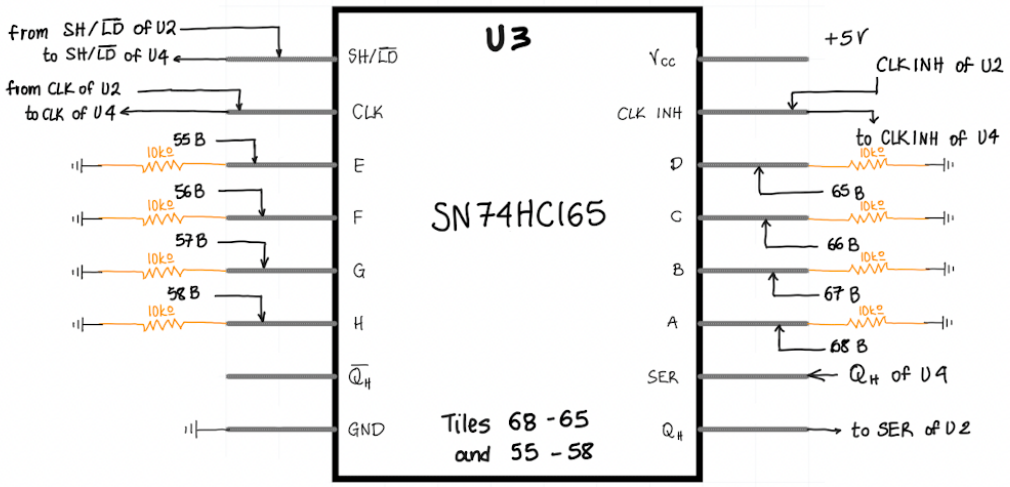


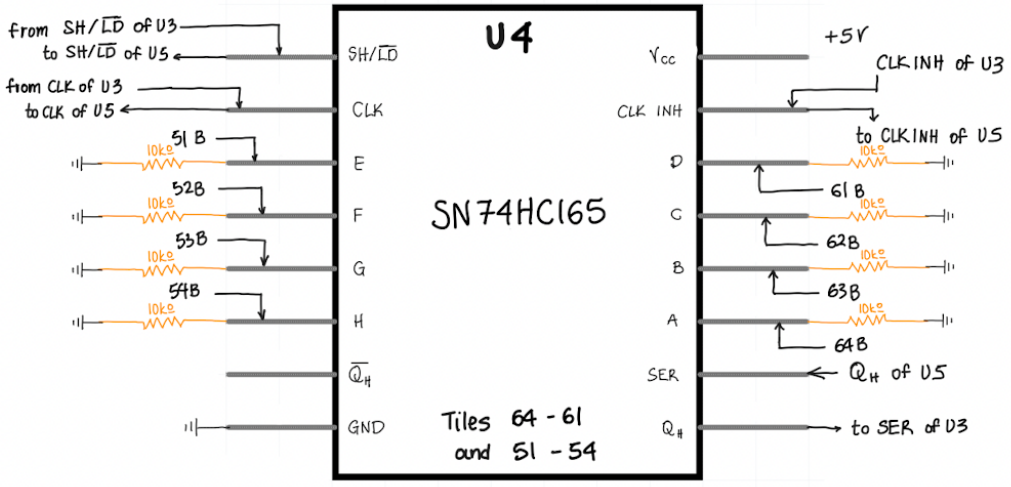
Circuit design:

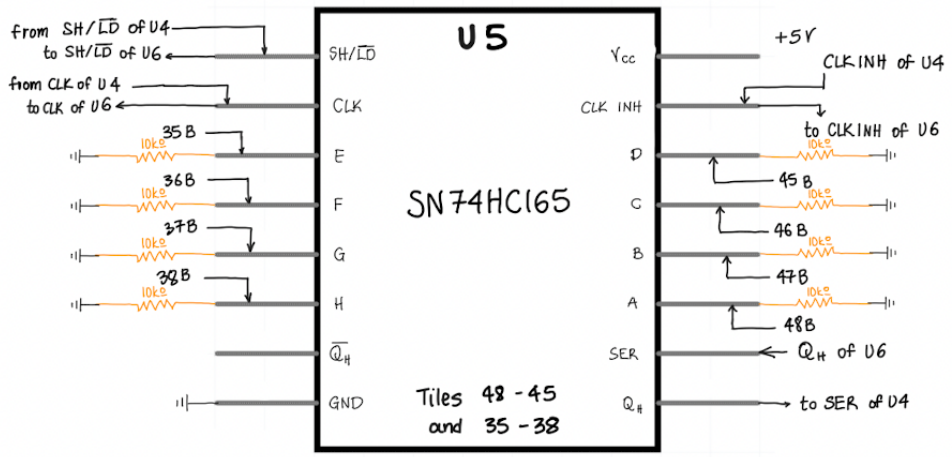

ARDUINO
ARDUINO MKR ZERO
USB Connection to PC/Serial Monitor
Battery Charger LED
Power
Li-Po 3.7 V
GND
PA08 SDA
PA09 SCL
PA21
+5V
LED_BUILTIN
+5V → Vcc
VIN
+3V3
GND
RESET
AREF/AIN[1] PA03 AREF
DAC0/AIN[0] PA02 A0 D15
AIN[10] PB02 A1 D16
AIN[11] PB03 A2 D17
AIN[4] PA04 A3 D18~
AIN[5] PA05 A4 D19~
AIN[6] PA06 A5 D20
AIN[7] PA07 A6 D21
to CLK of U1 ← PA22 D0~
PA23 D1~
from QH of U1 → PA10 D2~
PA11 D3~
to CLK INH of U1 ← PB10 D4~
PB11 D5~
D14 PB22 TX (SC5)
D13 PB23 RX (SC5)
~D12 PA09 SCL (SC2)
D11 PA08 SDA (SC2)
~D10 PA19 CIPO (SC1)
D9 PA17 SCK (SC1)
~D8 PA16 COPI (SC1)
~D7 PA21
~D6 PA20 → to SH/LD of U1
Ground
Power
LED
Internal Pin
SWD Pin
Other Pin
Digital Pin
Analog Pin
Default
Microcontroller's Port
ARDUINO.CC

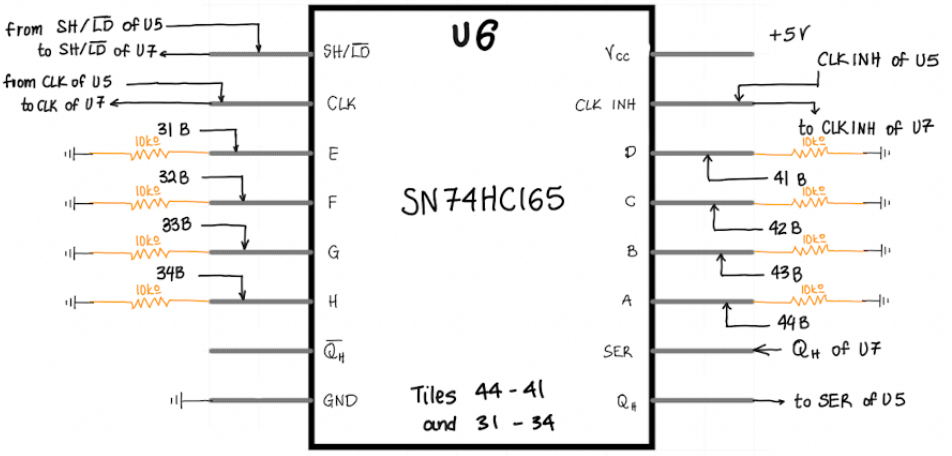

U1
SN74HC165
from pLoad Arduino pin
to SH/LD of U2
from clk pin of Arduino
to CLK of U2
SH/LD
CLK
E
F
G
H
QH
GND
75B
76B
77B
78B
10kΩ
Vcc
CLK INH
D
C
B
A
SER
QH
+5V
clken pin of Arduino
to CLKINH of U2
85B
86B
87B
88B
QH of U2
to data pin of Arduino
Tiles 88-85
and 75-88

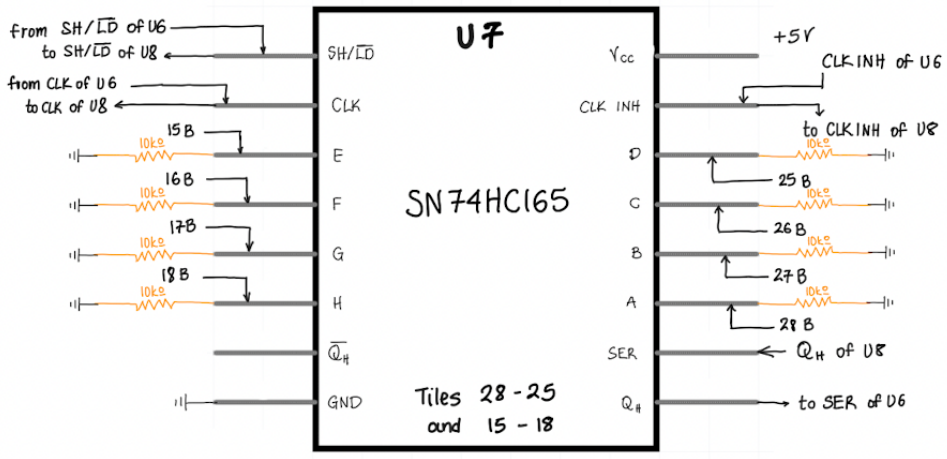

U2
SN74HC165
from SH/$\overline{LD}$ of U1
to SH/$\overline{LD}$ of U3
from CLK of U2
to CLK of U3
71B
72B
73B
74B
10kΩ
SH/$\overline{LD}$
CLK
E
F
G
H
$\overline{Q_H}$
GND
Tiles 84 - 81
and 71 - 74
V_{CC}
CLK INH
D
C
B
A
SER
Q_H
+5V
CLK INH of U1
to CLK INH of U3
81B
82B
83B
84B
Q_H of U3
to SER of U2

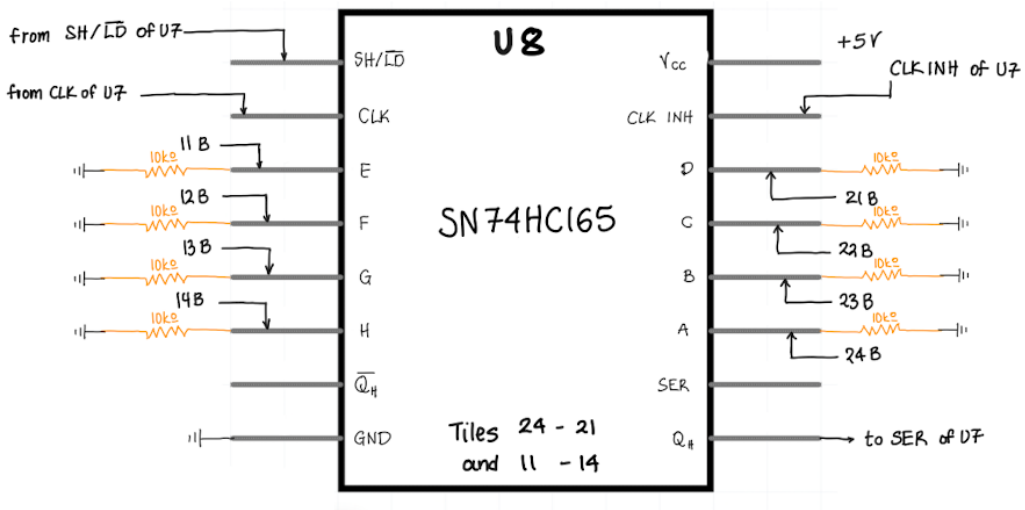
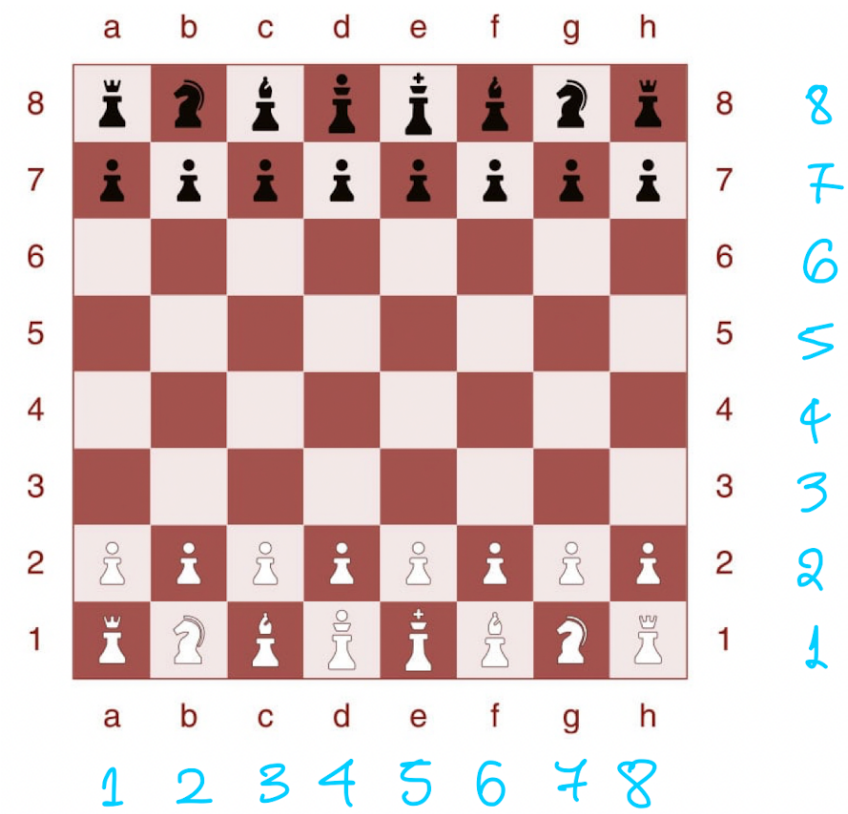

U3
SN74HC165
from SH/$\overline{LD}$ of U2
to SH/$\overline{LD}$ of U4
from CLK of U2
to CLK of U4
55B
56B
57B
58B
10kΩ
SH/$\overline{LD}$
CLK
E
F
G
H
$\overline{Q_H}$
GND
Tiles 68 - 65
and 55 - 58
V_{CC}
CLK INH
D
C
B
A
SER
Q_H
+5V
CLK INH of U2
to CLK INH of U4
65B
66B
67B
68B
Q_H of U4
to SER of U2


U4
SN74HC165
from SH/$\overline{LD}$ of U3
to SH/$\overline{LD}$ of U5
from CLK of U3
to CLK of U5
51B
52B
53B
54B
10kΩ
SH/$\overline{LD}$
CLK
E
F
G
H
$\overline{Q_H}$
GND
Tiles 64 - 61
and 51 - 54
V_{CC}
CLK INH
D
C
B
A
SER
Q_H
+5V
CLK INH of U3
to CLK INH of U5
61B
62B
63B
64B
Q_H of U5
to SER of U3


U5
SN74HC165
from SH/$\overline{LD}$ of U4
to SH/$\overline{LD}$ of U6
from CLK of U4
to CLK of U6
35B
36B
37B
38B
10kΩ
SH/$\overline{LD}$
CLK
E
F
G
H
$\overline{Q_H}$
GND
Tiles 48 - 45
and 35 - 38
V_{CC}
CLK INH
D
C
B
A
SER
Q_H
+5V
CLK INH of U4
to CLK INH of U6
45B
46B
47B
48B
Q_H of U6
to SER of U4


U6
SN74HC165
from SH/$\overline{LD}$ of U5
to SH/$\overline{LD}$ of U7
from CLK of U5
to CLK of U7
31B
32B
33B
34B
10kΩ
SH/$\overline{LD}$
CLK
E
F
G
H
$\overline{Q_H}$
GND
Tiles 44 - 41
and 31 - 34
V_{CC}
CLK INH
D
C
B
A
SER
Q_H
+5V
CLK INH of U5
to CLK INH of U7
41B
42B
43B
44B
Q_H of U7
to SER of U5


U7
SN74HC165
from SH/$\overline{LD}$ of U6
to SH/$\overline{LD}$ of U8
from CLK of U6
to CLK of U8
15B
16B
17B
18B
10kΩ
SH/$\overline{LD}$
CLK
E
F
G
H
$\overline{Q_H}$
GND
Tiles 28 - 25
and 15 - 18
V_{CC}
CLK INH
D
C
B
A
SER
Q_H
+5V
CLK INH of U6
to CLK INH of U8
25B
26B
27B
28B
Q_H of U8
to SER of U6



Chessboard numbering reference for code: