Charles Hong
Fall 2021

CS120B Final Project: Maze

Full Demo Link:
https://drive.google.com/file/d/1gdwfyb-8gH2mDGFBX-W8k5T85cCem0cX/view?usp=sharing

complexity 1:
https://drive.google.com/file/d/11ZW6Lu3F8vsDwNr4Jkd-ea_2P1ZiHGcE/view?usp=sharing

complexity 2:
https://drive.google.com/file/d/1cbfGRmzglm8k5zVxBTdjddRwTSf8hf61/view?usp=sharing

complexity 3:
https://drive.google.com/file/d/1gdwfyb-8gH2mDGFBX-W8k5T85cCem0cX/view?usp=sharing

Project Description:

This embedded system project simulates a game of maze. The maze is implemented on an 8x8 LED matrix, a joystick, along with EEPROM which saves the high score, as well as displaying custom characters onto the screen. The green dots represent the maze walls, while the singular red dot is the player. The objective is to have the player start from the top left corner and traverse to the bottom right corner with the joystick. The 2x16 LCD Screen is used to display the winning or losing messages with custom characters, as well as what the buttons do and the high score. The scoring system is basically the remaining time after completing the mazes. The more time you have left, the higher the score. The player doesn't get to see their score in the end however, they will only get to see high scores. The high scores are saved through EEPROM.

User Guide:

Rules:

1. You have 20 seconds to complete the two given mazes.
2. Starting from the top left corner, you have to reach the bottom right corner of both mazes in order to win the game.
3. You are allowed to reset and start the game whenever you'd like.
4. Player cannot go through walls.

Controls:

Left Button: Press to start the game. The timer will start counting down.
Right Button: Press to reset the game
Joystick: Move left, right, up, down, and the red dot will move once to the respective direction.

Special Considerations:

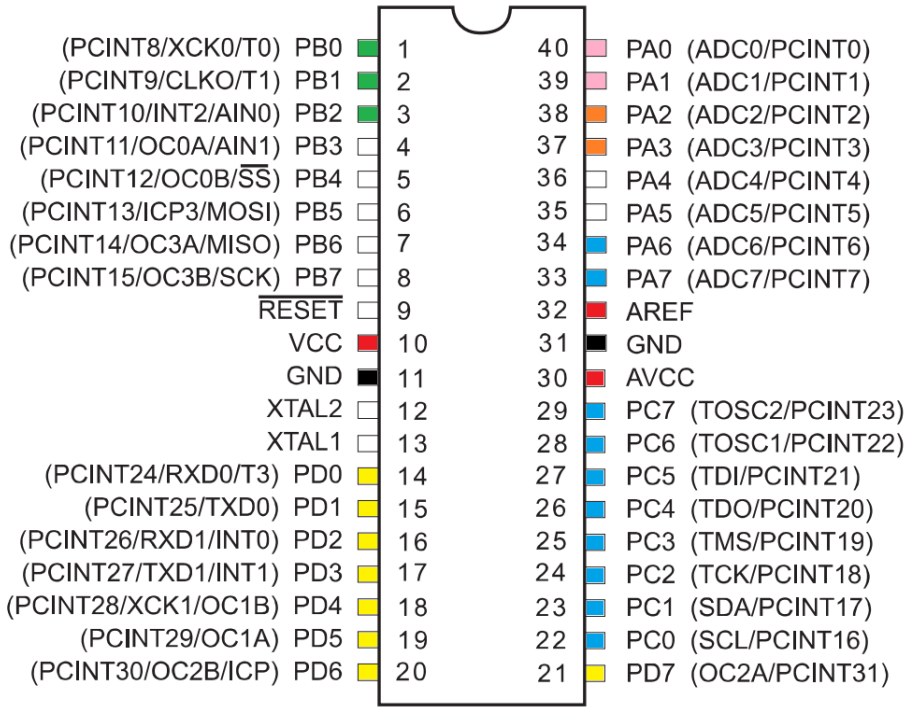
Due to the LCD Display taking some time writing to the screen, you can see that the LED Matrix sometimes flashes a bit before settling down. This is because the Matrix is refreshing at such a fast rate that the human eye thinks it is constantly outputting all LEDs at once, but really it is just displaying one row at a time. But because the LCD Write to screen process is long, it will delay the LEDs from showing at a constant rate, so it slows down the refresh rate and starts blinking.

Source File Description:

- chong039_phase3.c - Contains code for EEPROM, LCD Custom Characters, Joystick controls, and LED Matrix Display through Shift Registers.
    - EEPROM Code - avr-libc: <avr/eeprom.h>: EEPROM handling (nongnu.org)
    - LCD Custom Character - Display Custom Character on 16X2 LCD to AVR Microcontroller (Atmega-8) « Hack Projects (wordpress.com)
    - ADC Initialization code - EECS120B Lab 08 - A2D (light meter) - Google Docs
    - ADC Channel Switching - ATmega1284 (microchip.com)
    - Learned how to use a Shift register from here - Using a Shift Register to Control Multiple LEDs | Onion Omega2 Maker Kit
    - Task Scheduler - EECS120b Lab 11 - scheduler (Scrolling game) - Google Docs
    - Timer - EECS120b Lab 06 - Timer synchSMs (Reflex game) - Google Docs
- io.h - Standard LCD Display Header file, **Code obtained from EECS120b Lab 07 - LCD - Google Docs**
- io.c - Standard LCD Code from lab 7. The only difference is that instead of using PORTD, I am using PORTA. **Code obtained from EECS120b Lab 07 - LCD - Google Docs**

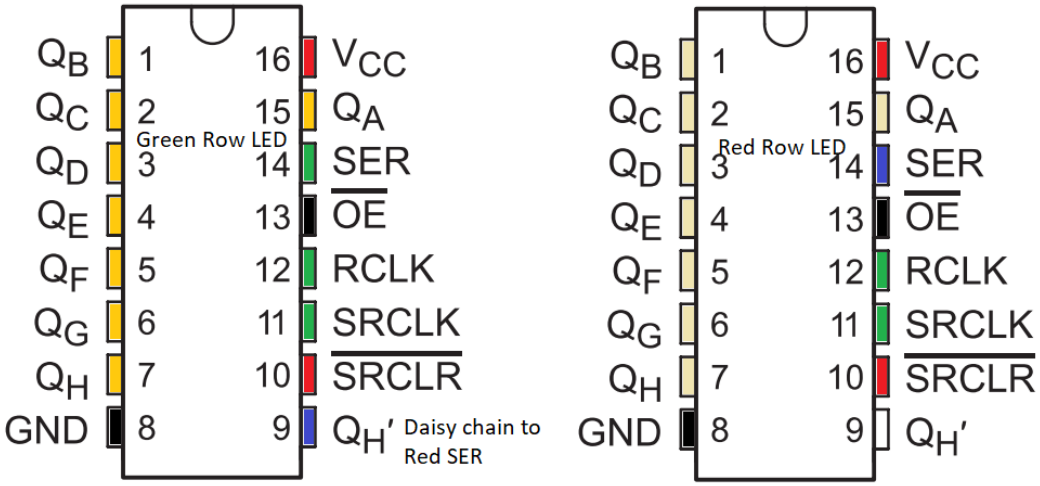
Component Visualization

ATMEGA1284



Pink: 2 pins for 2 channels of ADCs for the joystick. PA1 is for up and down, PA0 is for left and right.
Orange: Two Buttons, L and R. L goes to PA4. R goes to PA3.
Red: Connects to 5V.
Black: Connects to Ground.
Blue: Connect to the pins on the 2x16 LCD Display.
Green: PB0-PB2 connects to the Green Row Shift Register. PB0 is SER. PB1 is RCLK. PB2 is SRCLK.
Yellow: Due to lack of shift registers, PDn is used for the Column selection. It is directly connected to the LED Matrix that lights up the columns.

Shift Registers



Green Row LED Shift Register:

- QA-QH: Connects to the Green LED Matrix Row.
- SER: Connected to PB0.
- RCLK: Connected to PB1.
- SRCLK: Connected to PB2.
- QH': Connects to Red Row LED SR's SER (pin 13).

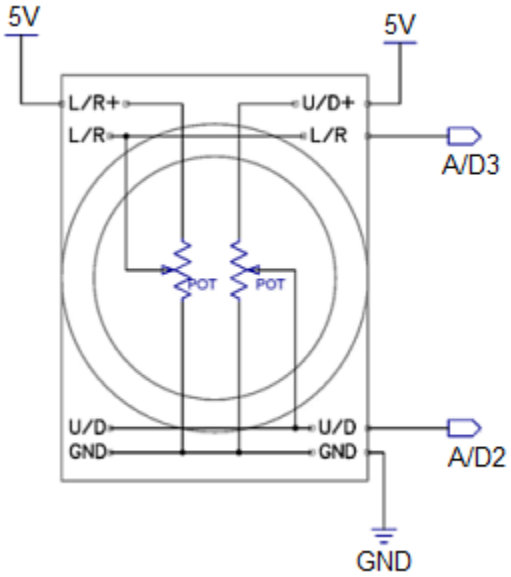
Red Row LED Shift Register:

- QA-QH: Connects to the Red LED Matrix Row.
- SER: Connected to PB0.
- RCLK: Connected to PB1.
- SRCLK: Connected to QH' from Green Row LED SR for daisy chaining.

SRCLR, VCC: Red, connects to 5V

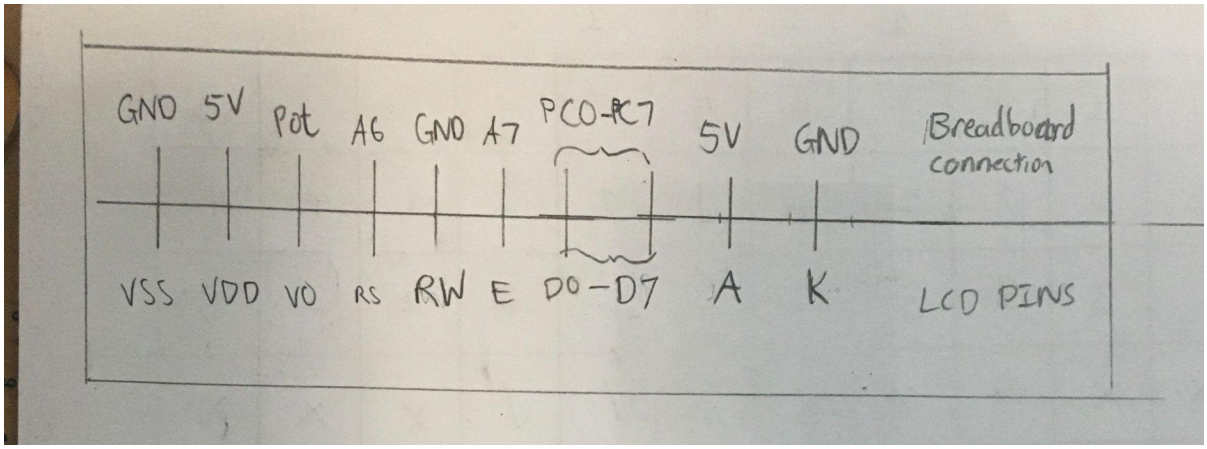
GND: Black, connects to ground.

Joystick

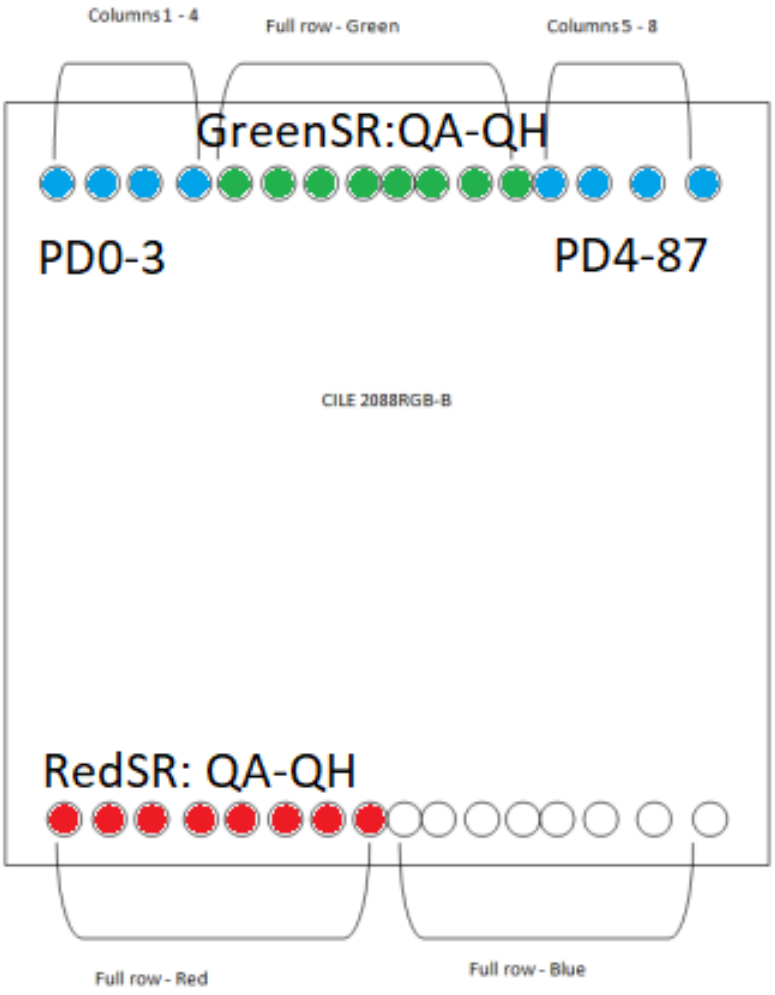


A/D2 or U/D is connected to PA1.
A/D3 or L/R is connected to PA0.

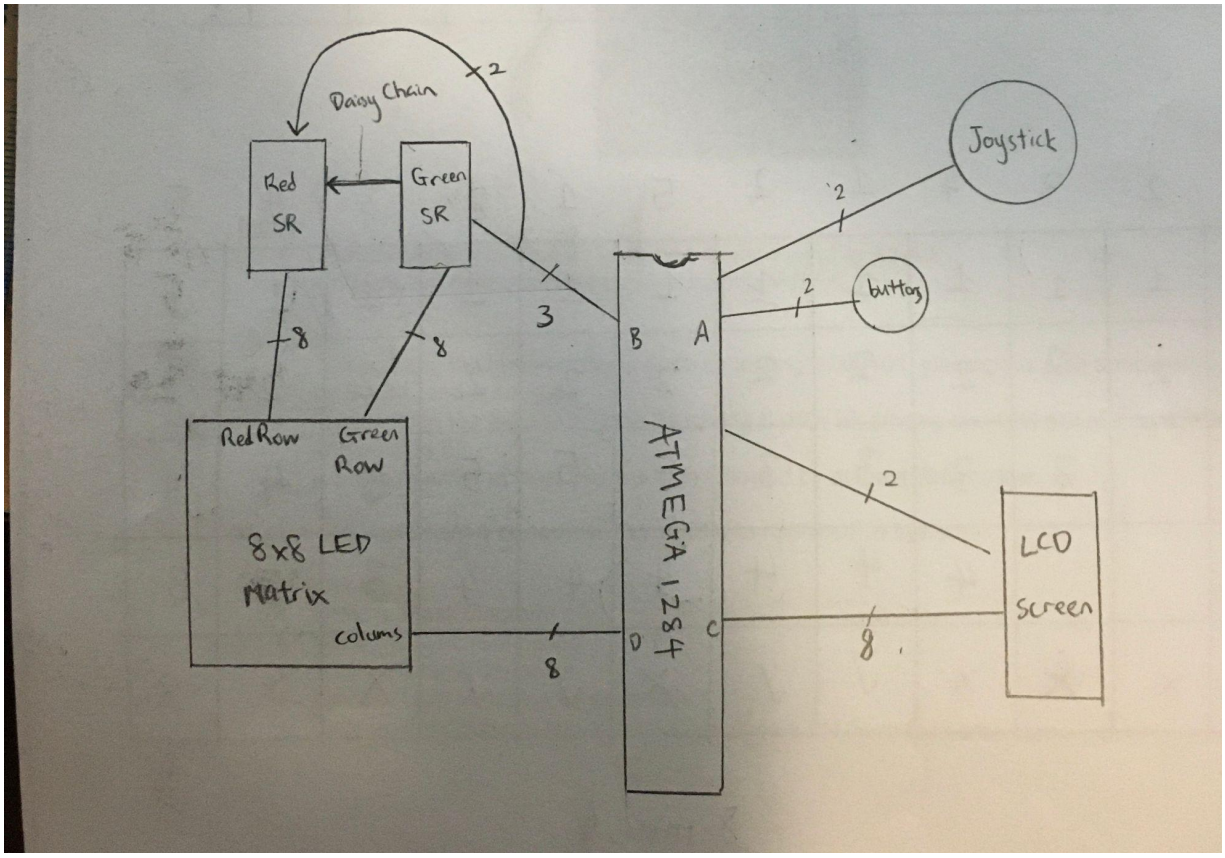
2x16 LCD Display

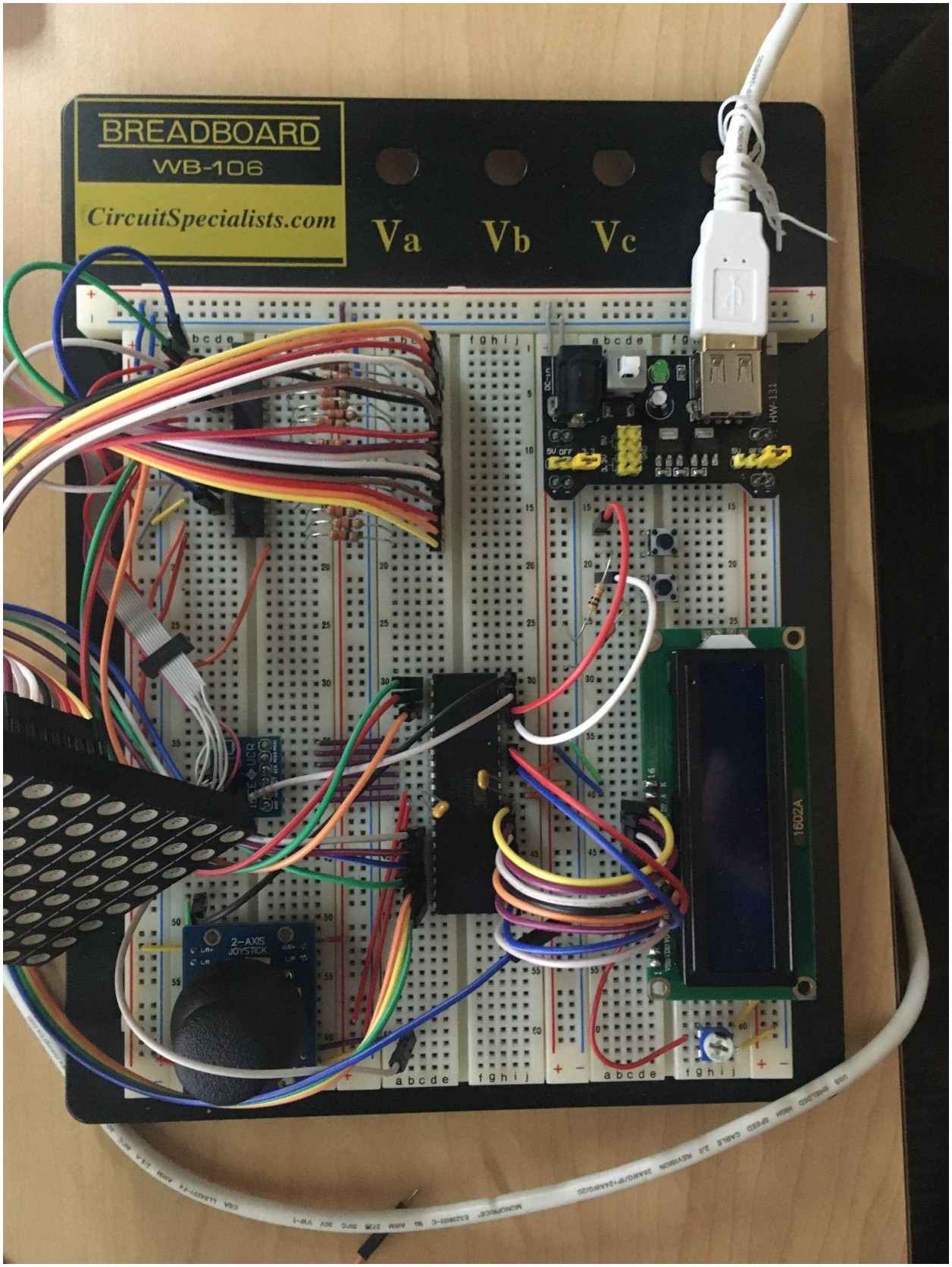


8x8 LED Matrix Common Anode



Overhead Layout




BREADBOARD
WB-106
CircuitSpecialists.com
Va
Vb
Vc
HW-131
2-AXIS
JOYSTICK
1602A

Technologies Learned

Learned how to write and store custom characters in the LCD Display.
Learned how to save and load data through EEPROM.
Successfully utilized a joystick by switching the analog to digital channels in port A with ADMUX.
Effectively output more bits of data with less pins by using the shift register. This is very helpful if there are not that many available pins on the microcontroller.
Successfully displayed images onto an LED Matrix through refreshing the rows and columns at a very fast rate.

External Resources

8x8 LED Matrix Pinout - Using an 8×8 RGB Matrix with Arduino – The Raspberry Blonde (wordpress.com)

Joystick Pinout - Joystick Wiring & Example Code | LEARN.PARALLAX.COM

Shift Register Datasheet - SNx4HC595 8-Bit Shift Registers With 3-State Output Registers datasheet (Rev. J) (ti.com)

LCD Custom Character - Display Custom Character on 16X2 LCD to AVR Microcontroller (Atmega-8) « Hack Projects (wordpress.com)

Learn how LED Matrix works - 8x8 LED Matrix for Arduino - YouTube

EEPROM, ADC - ATmega1284P Datasheet (microchip.com)