Report

Full Demo Link: https://www.youtube.com/watch?v=imeUPkgvfZs&t=2s

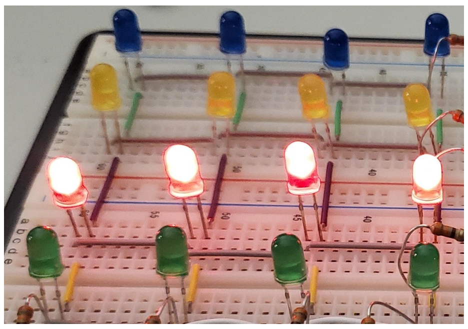
Project Description

The project I created is a memorization game that consists of three complexities. A Joystick, LED matrix with a shift register, and a Nokia 5110 screen. The nokia screen is used to display the menu to the player and different options they can select. Once the game starts the screen will output the current level and score. The LED matrix is used to display the different LEDs that turn on/off by using the shift register and the Joystick is used to navigate the LED matrix.

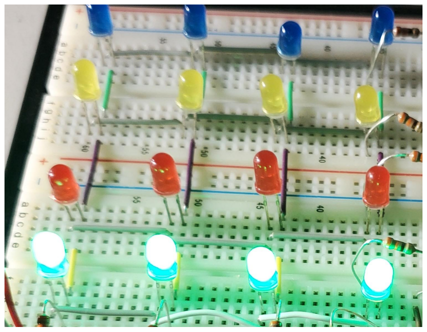
User Guide

Rules: The point of the game is to memorize the lights that individually appear after a row of green lights. Level 1 consists of 1 LED to memorize then Level 2 has 2 LED and level 3 has 3 LED to memorize. I only created 3 levels to keep the demo video short but more levels could easily be added. If the Player is able to clear all three levels then a victory screen appears. If the player selects a wrong LED in any of the 3 levels then a row of red LEDs will appear with a message saying “You lose”.

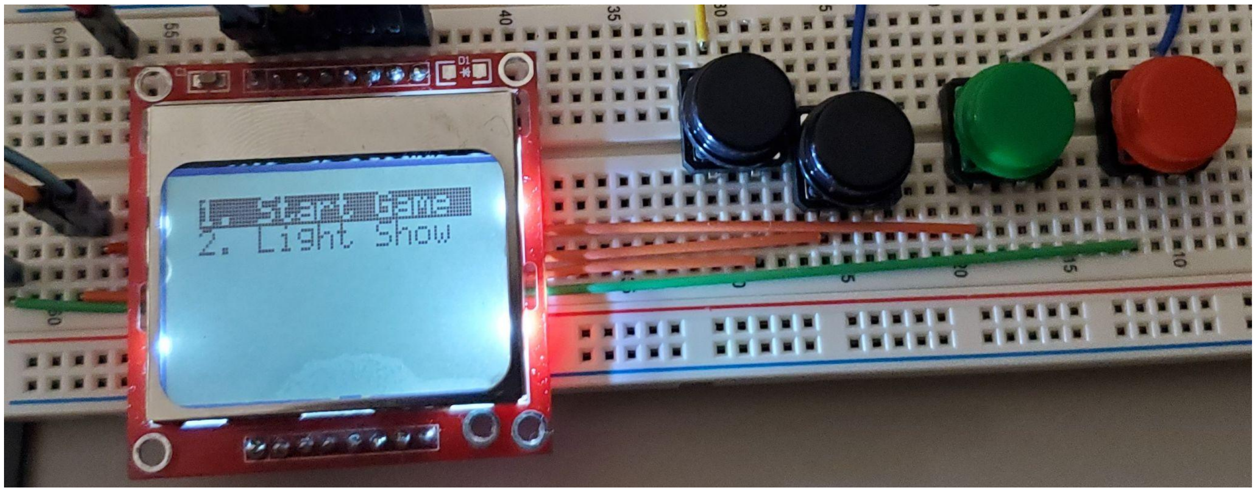
Wrong Input



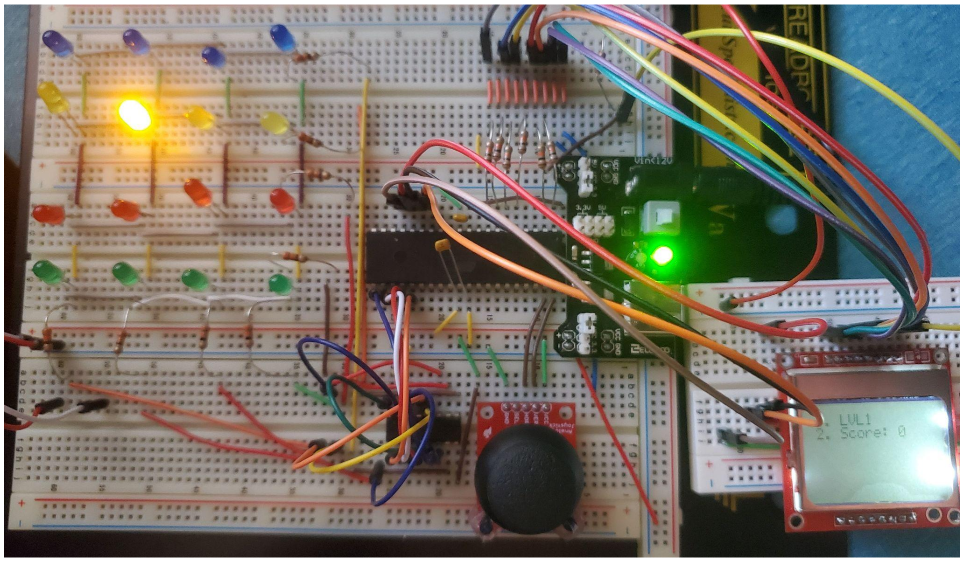
Right Input



Controls:
- (**left**) **Black button-** Moves up on menu screen
- (**right**) **Black button-** Moves down on menu screen
- **Green button-** Is used for selecting an option in the menu and is used for selecting LEDs on the Led Matrix once the game starts.
- **Red button-** is used for restarting the game and if the user loses during the game. The button is only active once the game starts. It has no effect on the menu screen.



-**JoyStick-** Once the Start Game is selected then the Joystick is only used for the LED matrix to navigate through the different LEDs. The joystick has no effect on the starting menu.



Source File Links:

ADC.h - Used to initialize ADC ports PB0 & 1 and get the voltages from the joystick

JoyStick.h - Contains function XY_Position that based on the PB0(x) and PB1(y) values that are passed generates values from 0-1023 that will determine if the joystick was move ↑ ↓→← and the diagonals. From this a 4X4 matrix that contains the LED position is adjusted accordingly based on the joystick movement

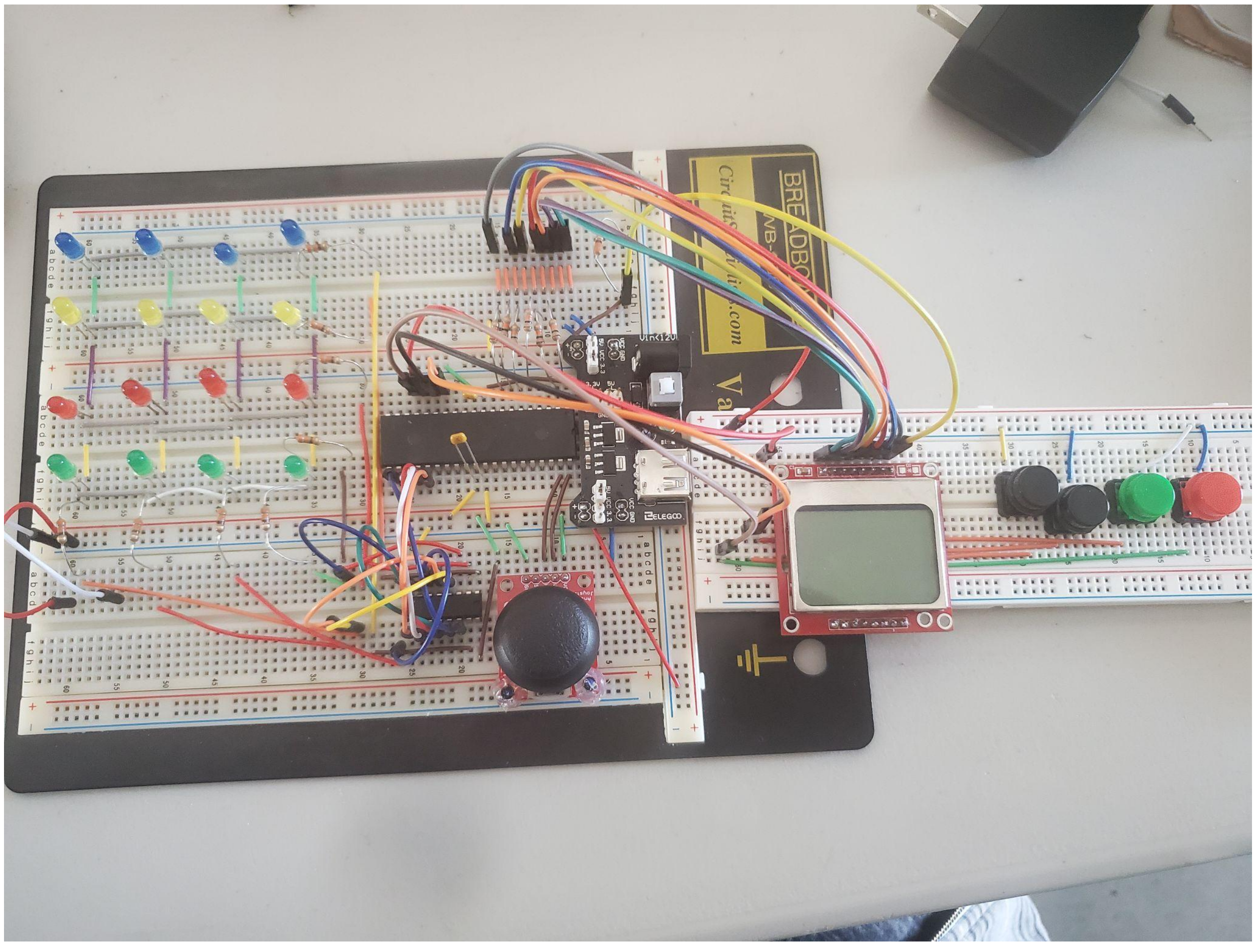
font.h - contains a library of letters of each pixel that corresponds to the screen in reverse. I got it from https://www.avrfreaks.net/forum/c-display-pcd8544-displaying-degree-character-%C3%A2%C2%B0-qu I added a sixth column so that there is a space between the letters that are ouputed.

Nokia.h - Contains the functions for initiating SPI in the microcontroller to output each bit from font.h to the nokia screen. It also Contains functions in the .h file that describe in much more detail the initialization of the screen to specific specifications and how to clear the nokia screen.
timer.h - used to generate the proper time delay triggered by TimerFlag in main.c
simAVRHeader.h - used to simulate the project

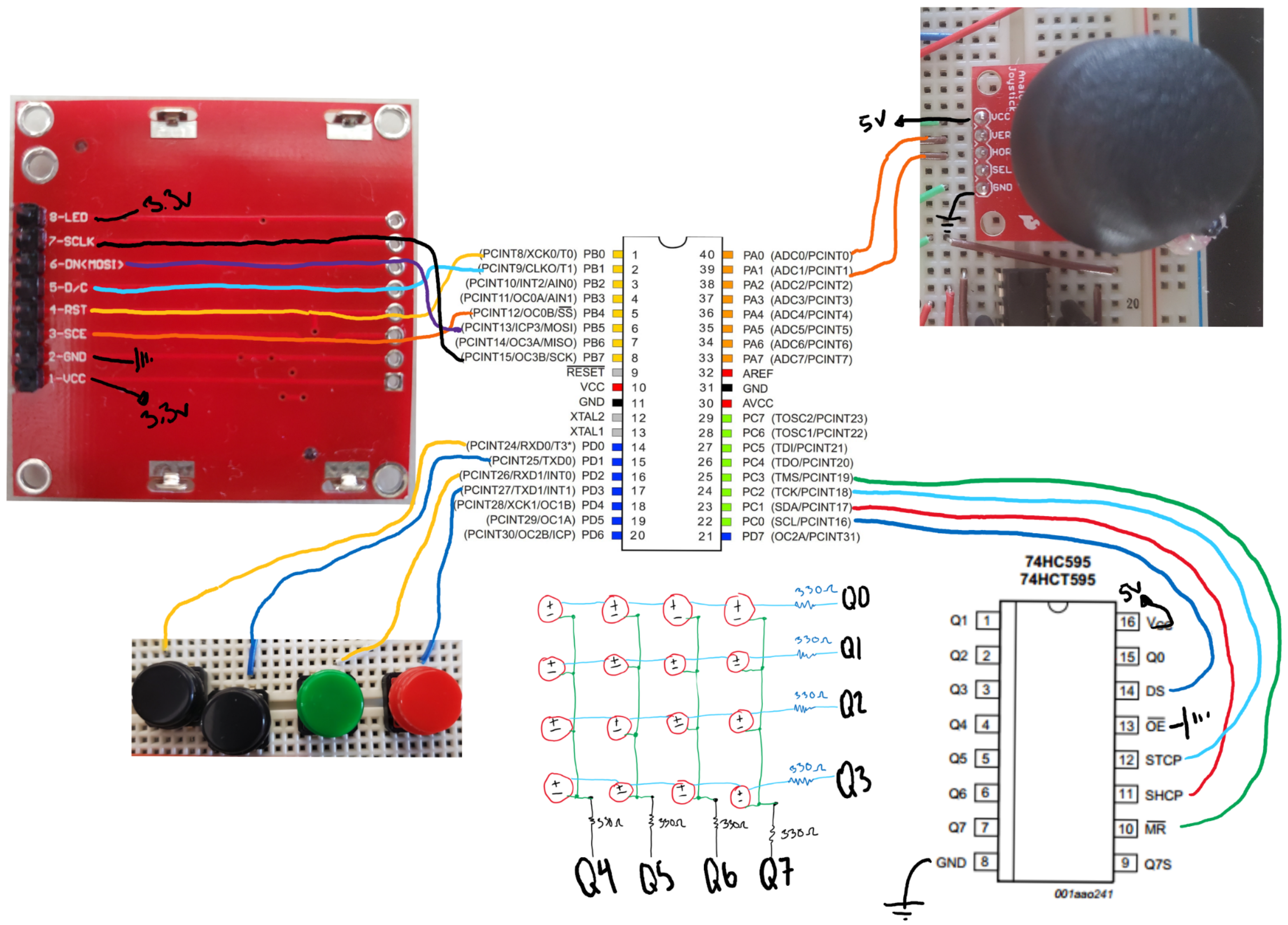
main.c - Contains 3 main states

-**JoystickState** - This state is used to detect the movement of the joystick once the game starts and updates the LED matrix.
-**NokiaState** - The state is used to output information such as the menu, score, level, winning/losing screen.
-**GameState** - contains the logic of the game such as the rules and win/losing conditions.

Overview



Component Visualization:



Technologies Learned

I learned how to send data to other devices using SPI and MOSI, and also receive data using ADC input. I also gained an understanding on how to implement and use shift registers to save pins on the microcontroller. After doing this project I am now comfortable looking at other microcontrollers and beable to utilize them with new technologies such as a motion sensor or temperature sensor by reading the data sheet and turning on the correct bits. I plan to continue and expand my project to implement more functionality and learn more features about the atmega1284.