ELEC 327 Tic Tac Toe Project

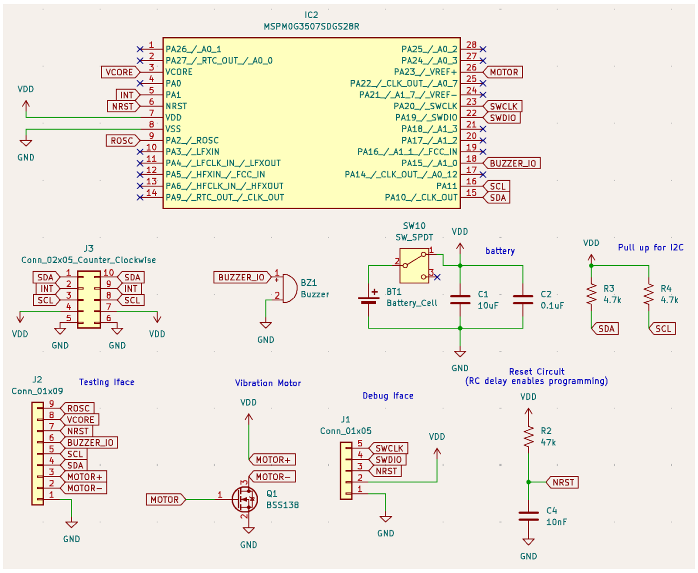
Lance Yoshizaki, Guido De Santis

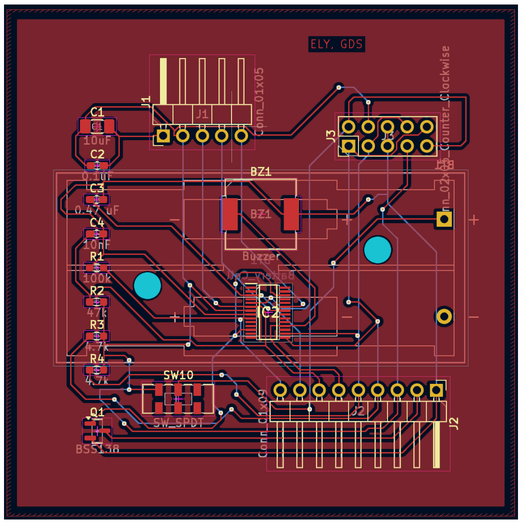
Concept

The goal of this project is to create a Tic Tac Toe game using the Adafruit NeoTrellis RGB LED Keypad while also driving audio and haptic feedback, all controlled by a custom-designed PCB. The project aims to use the MSPM0's I2C communication lines to interface with the keypad as well as GPIO functions to drive the buzzer (audio) and vibration motor (haptic). The PCB houses a battery cell, aiming to take advantage of the MSPM0's low-power requirements.

Design

Schematic + PCB designs:





Part of the design was heavily influenced by Lab 5 and Simon PCBs, especially the battery cell (and corresponding capacitors), reset circuit, debug interface, and buzzer.

The datasheet was used to determine the choice of pins driving the motor circuit as well as determining the SCL and SDA pins to interface with the keypad.

The layout of the PCB was designed to minimize the amount of trace on the board, as we placed the MCU at the center of the board. We also made sure that soldering would be easier (after experience from lab 8) by making sure that traces would not be too close to soldering pads. The battery cell went on the other side of the board to minimize the area, which came out to be 2.6 in x 2.6 in.

Implementation

We were able to reduce the number of MCU pins to use by using the NeoTrellis keypad, as individually driving many LEDs would take too many GPIO pins even if we used LED drivers to try to mitigate this. We were thus able to choose the MSPM0G3507SDGS28R which only has 28 pins but is easier to solder compared to other packages with more pins.

The vibration motor is essentially a small motor that rotates very fast to create a vibrating effect inside a small compartment. The datasheet provides the below table:

Parameters	**Values**	**Units**
Input Voltage	3.7	V_{DC}
Operating Voltage Range (DC)	3.0 ~ 4.5	V_{DC}
Starting Voltage	2.5	V_{DC}
Direction of Rotation	CW	-
Rated Speed Rated Voltage, Rated Load, Motor Fixed	8,000 ± 3,000	RPM
Rated Load Current Maximum at 3.7 V_{DC}, Rated Load, Motor Fixed	85	mA
Starting Current Maximum at 2.5 V_{DC}, Rotor Locked	170	mA

We see that 3.3 V is well within operating voltage range , but we need to provide a starting current of 170 mA to it which is too high for what a MCU pin can produce. Thus, we used a N-Channel MOSFET to produce a sufficient current (up to 220 mA).

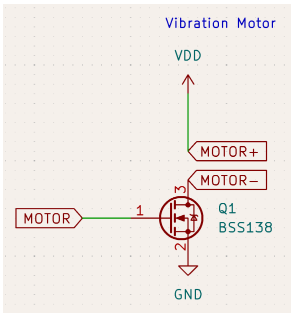
I_S	Maximum Continuous Drain–Source Diode Forward Current	–	–	0.22	A

We made sure we could switch the MOSFET with 3.3V logic by identifying the gate threshold voltage was below 3.3 V.

ON CHARACTERISTICS

$V_{GS(th)}$	Gate Threshold Voltage	$V_{DS} = V_{GS}$, I_D = 1 mA	0.8	1.3	1.5	V

The buzzer and vibration motor are driven simultaneously using similar methods of switching the GPIO pins HIGH and LOW, but we maintain a roughly constant driving frequency for the vibration motor independent of the frequency of the buzzer.



Circuit driving the vibration motor

The Adafruit NeoTrellis uses I2C communication to operate, so we used the pins designated to SCL and SDA functionality on the MSPM0 to interface with the device. We attempted to adapt the TI-provided driverlibs to interface with the I2C device, but was not able to control the LEDs on the NeoTrellis board. Adafruit provides libraries in C++ that are much easier to use so we produced a proof of concept to demonstrate the game logic by using an Arduino to communicate with the keypad.

Game Logic

1) When the game starts, a 3x3 Matrix (top left when wires stick out from the top) of the 4x4 grid lights up by calling the Wheel() function from the Seesaw library, which generates smooth color gradients across the RGB color wheel. The function works by taking a value from 0 to 255 and returns a corresponding RGB color. In this case it generates a rainbow effect on the NeoTrellis that goes from left to right.
2) The first player (represented by red leds) presses a button in the 3x3 matrix
3) The second player (represented by blue leds) presses another button in the matrix
4) If a player succeeds in placing three of their colored LEDs in a row — horizontally, vertically, or diagonally — they win, and all LEDs in the 3×3 matrix flash in that player's color three times. If all buttons in the matrix have been pressed and no player has won, the game ends in a draw and the 3×3 LEDs flash white three times.
5) Pressing any button in the matrix after a win or draw resets the game and restarts the rainbow animation.

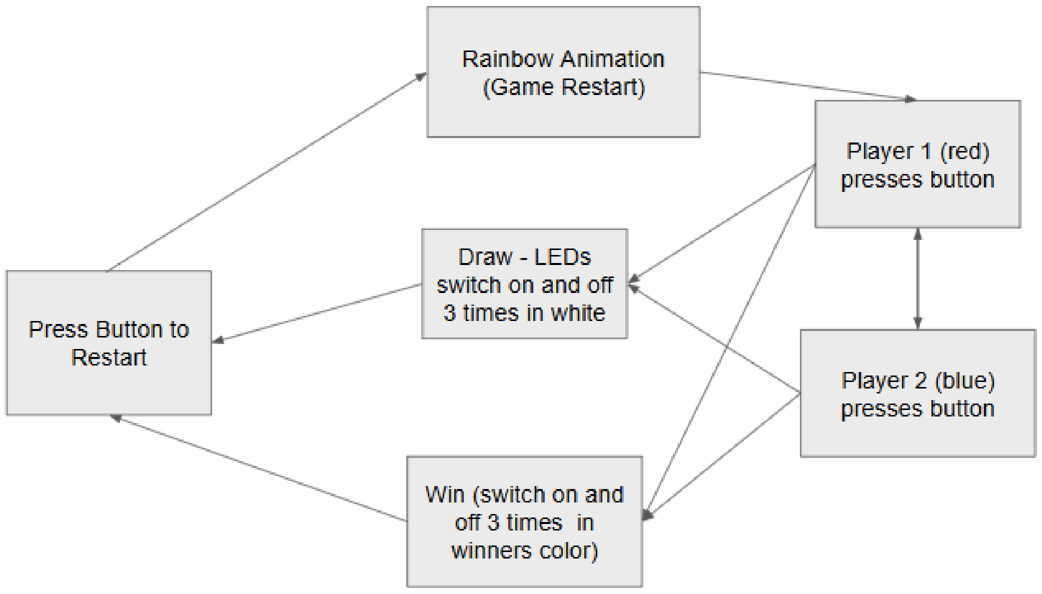

Rainbow Animation (Game Restart)
Player 1 (red) presses button
Draw - LEDs switch on and off 3 times in white
Press Button to Restart
Player 2 (blue) presses button
Win (switch on and off 3 times in winners color)

Diagram representing the State Machine

Videos + Github

Demo:
https://drive.google.com/file/d/1fjA7qsOB4qs4aa6QqQU2F6MItRrnvWFh/view?usp=sharing

Presentation:
https://drive.google.com/file/d/1UJASU8I_02HgeJsb4cwUk8FmOSuKSuBJ/view?usp=sharing

Github Repo: