# CAB202 Assignment:

Student Name: Mattias Winsen

Student Number: n10467874

TinkerCAD Link: <https://www.tinkercad.com/things/eLugW85uI3S-fantastic-bojo/editel>

Video Demonstration Link: <https://youtu.be/5Dui570ZQ0s>

## Introduction

A game, heavily based on the card game ‘Snap’, has been designed to satisfy the provided requirements. The game begins by displaying a very simple UI on the LCD, which displays a user’s score (first to zero wins) along with the current card. Players can then press their push buttons, providing analogue input to the Arduino, to ‘flip’ to flip their card (like flipping a card when playing snap). A number associated with that card is then displayed on the LCD. The user can then interact with with a potentiometer (which is used as a placeholder analogue input) to ‘snap’ when they think there have been two of the same numbered cards in a row. Ideally, the potentiometer would be replaced by a pressure sensor, that way the user must hit the sensor with a defined amount of force to trigger a snap. Unfortunately, Tinkercad didn’t have a pressure sensor, and as such a potentiometer was used as a placeholder. In a real world application, this could be replace by a pressure sensor such as the one found [here](https://au.element14.com/smartec/spd030g/pressure-sensor-0-200kpa-dip-6/dp/2543409?gclid=Cj0KCQjwoJX8BRCZARIsAEWBFMIjWFxbzLGBTYANVbcLZFqUXM9xUptIEK1By3f6CYPHLr5wJRB5_PcaAo7dEALw_wcB&mckv=s_dc|pcrid|432287894049|pkw||pmt||slid||product|2543409|pgrid|102205442564|ptaid|pla-900340835440|&CMP=KNC-GAU-GEN-SHOPPING-SEMICONDUCTOR-ICS). If the user has snapped at the right time, a LED associated with them will light up (digital output) and their oppositions score will be reset to the starting score. If they fail to snap at the right time, their score will be reset to it’s starting value, and a buzzer will play a tone (control by PWM) to reinforce their error. Additionally, when snapping, the user’s reaction time is logged to serial (using UART to log and a timer to calculate the reaction time) for an additional challenge

|  |  |
| --- | --- |
| Digital I/O - Switch | Button 1 and 2 which are used for player’s input, specifically, ‘Flipping’ their card to display |
| Digital I/O – Debouncing | Debouncing was implemented in the form of a timer based debounce, which debounces button 1 and 2 |
| Digital I/O – LED | LED 1 and 2 are used to validate players when they have correctly ‘snapped’, lighting up accordingly when the snap on consecutive numbers |
| Analog Input – ADC | Potentiometer 2 and 3 are used as analogue input for when either of the users have ‘snapped’. Note, ideally, these would be replaced by 3 pin pressure sensors, so the user could hit them, however; due to limitations in Tinkercad (the software doesn’t have pressure sensors, or a reasonable replacement), potentiometers were used as placeholders. |
| Analog Output – PWM | The buzzer uses PWN to vibrate at different frequencies (and therefore different pitches) when a user snaps on a pair of non-matching consecutive numbers |
| Serial I/O – UART | UART is used in conjuncture with a timer to communicate a user’s reaction time in the serial log. |
| LCD | The LCD is used as the main display of the game, displaying the number of cards a player has left, along with the current ‘flipped’ card |
| Timers (other than debouncing or PWM) | A time is used for debouncing, PWM, and to record the reaction time of users in seconds (finding the time between when a number is displayed, and when a player presses their ‘snap’ button (potentiometer in the diagram) |

## Schematic

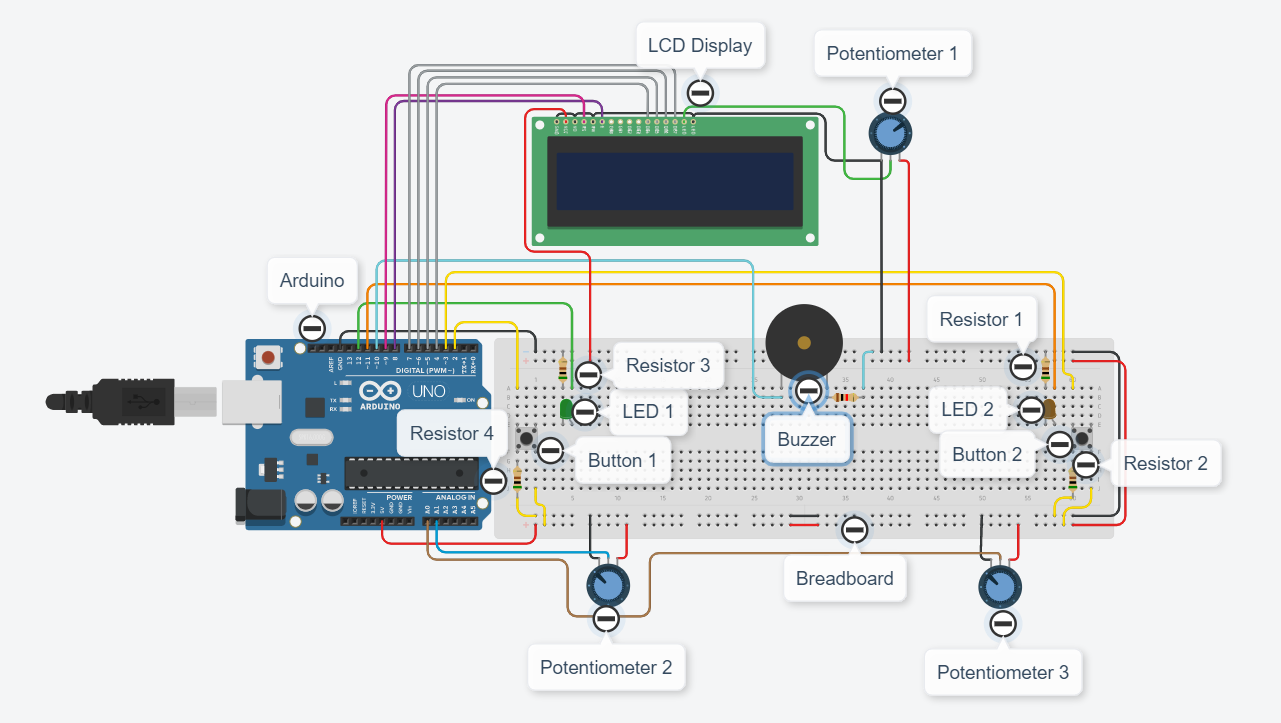


Figure - Labled Schematic

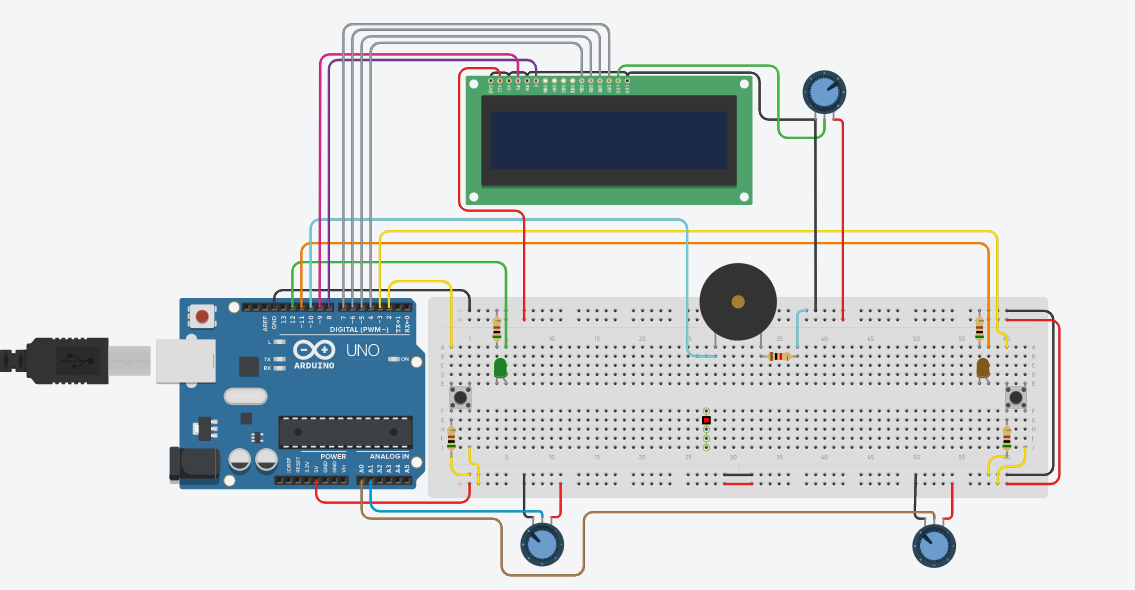


Figure – Un-labled Schematic (for ease of viewing)

## Wiring Instructions

1. To begin, wire the ground and 5V pin from the Arduino to the positive and negative rails of the breadboard, making sure to wire to both the top and bottom rail
2. Next, plug a button into one end of the breadboard, jumping it across the void in the middle. Then wire the bottom two pins to the positive and negative rail, accordingly, placing a 500 Ohm resistor in between the button and the negative rail. Wire one of the top pins of the button to pin 2 on the Arduino (Button 1\_.
3. Repeat this step for another button at the other end of the breadboard, wiring it to pin 3 on the Arduino (Button 2).
4. In a free space on the breadboard, plug in a buzzer and wire one end into a negative rail. Then, wire the other end into pin 10 of the Arduino
5. After that, wire a coloured LED in series with a 500 Ohm resistor at one end of the breadboard, running the Cathode to the negative rail, and the anode to pin 12 on the Arduino (LED 1).
6. Repeat this step at the other end of the breadboard, using a different coloured LED and wiring it to pin 11 (LED 2).
7. Wire a potentiometer, placed at one end of the board, so that the outer (power) pins run to the positive and negative rails, and so that the middle (wiper) pin runs to pin A0 on the Arduino (Potentiometer 2).
8. Repeat step 7 at the other end of the breadboard, wiring the wiper to pin A1 (Potentiometer 3)
9. Run a wire from pin VCC on the LCD to the positive rail of the breadboard
10. Wire pin GND, VO, RW, and the outer LED pin to the ground rail of the bread board
11. Run a wire from pin RS on the LCD to pin 9 on the Arduino
12. Run a second wire from pin E on the LCD to pin 8 on the Arduino
13. Then, wire pin DB4, DB5, DB6, DB7 on the LCD to pin 4,5,6 and 7 on the Arduino accordingly
14. Lastly, wire the inner LED pin on the LCD to the middle (wiper) pin on a potentiometer. Then wire the outer pins of the potentiometer to the positive and ground rails on the breadboard.