Using 16x8 LCD without I2C in TinkerCAD

Project Overview

This project demonstrates how to control a Liquid Crystal Display (LCD) 16x2 to show static text ("Hello World!") and a dynamically incrementing number. The rate at which the number increments will be controlled by a Potentiometer, allowing for variable speed adjustment. The entire circuit will be designed and simulated within the TinkerCAD Circuits environment.

Objectives

The primary objectives of this project are to:

• Familiarise with the TinkerCAD Circuits interface for hardware design and simulation.

• Understand the basic connections and operation of an Arduino Uno development board.

• Learn how to interface and control a LCD 16x2 display.

• Understand the function and wiring of a Potentiometer as an input device.

• Implement a time-based counter with adjustable intervals using the millis() function, avoiding the blocking delay() function.

• Utilise Serial Monitor for debugging and outputting data.

Components Required

To build this circuit in TinkerCAD, you will need the following components:

• Arduino Uno R3: An open-source development board that serves as the "brain" of the system, storing and executing the code to control the circuit.

• Breadboard (Small): A rectangular plastic board with tiny holes used to easily insert and connect electronic components without soldering. It consists of electrically connected rows and columns.

• LCD 16x2 Display: An alphanumeric character display with 2 lines and 16 characters per line, commonly used to show status and parameters in electronic applications.

• Potentiometer: A three-terminal electrical device with a knob used to control speed, voltage, or frequency by varying resistance. It works similarly to a voltage divider.

• Resistors:

◦ 220 Ohm Resistor: Typically used for the backlight of the LCD to limit current and prevent damage.

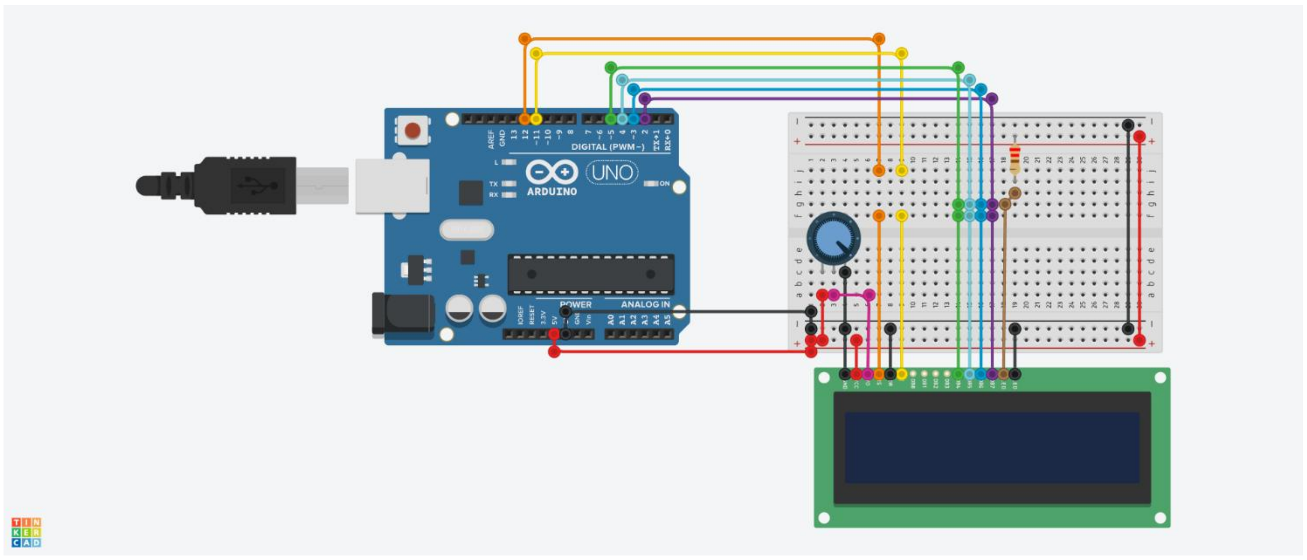
◦ 10k Ohm Potentiometer (for LCD Contrast): If you opt for adjustable contrast on the LCD, a potentiometer can be used; otherwise, a fixed resistor or direct connection to ground can be used. The primary potentiometer in this project controls the counter speed.

• Jumper Wires: Used to make electrical connections between components on the breadboard. It is good practice to color-code wires (e.g., red for positive, black for negative) for organization.

Circuit Assembly (Wiring Instructions)

Follow these steps to wire your circuit in TinkerCAD:

1. Start a New Circuit in TinkerCAD:
- Navigate to Tinkercad.com and sign in.
- From your dashboard, click on "Circuits" in the left menu.
- Click "Create new Circuit".
- Change the default random name to something descriptive, like "LCD Potentiometer Counter".
2. Add Components:
- From the "Components" menu on the right, drag an Arduino Uno R3 and a Breadboard Small onto the workplane.
- Rotate them (using the rotate tool in the upper left) for a convenient layout, e.g., portrait orientation.
- Add an LCD 16x2 display and a Potentiometer.
- Add any necessary resistors (e.g., 220 Ohm for LCD backlight, and optionally a 10k Ohm potentiometer for LCD contrast).
3. Power the Breadboard:
- Connect the Arduino's 5V pin to the positive (+) power rail of the breadboard (use a red wire).
- Connect the Arduino's GND pin to the negative (-) ground rail of the breadboard (use a black wire).
- If using a half-sized breadboard where the buses are not connected across the middle, run jumper wires to connect the positive rails on both sides and the negative rails on both sides.
4. Connect the LCD 16x2 Display:
- Pin 1 (GND): Connect to the breadboard's negative (-) ground rail.
- Pin 2 (VCC): Connect to the breadboard's positive (+) power rail.
- Pin 3 (VEE - Contrast): Connect to the wiper (middle pin) of a 10k Ohm potentiometer. Connect one outer pin of this potentiometer to the positive (+) power rail and the other outer pin to the negative (-) ground rail. This allows you to adjust the LCD contrast. Alternatively, for fixed maximum contrast, connect this pin directly to the negative (-) ground rail.
- Pin 4 (RS - Register Select): Connect to Arduino Digital Pin 7.
- Pin 5 (RW - Read/Write): Connect to the breadboard's negative (-) ground rail (to set it to write mode).
- Pin 6 (E - Enable): Connect to Arduino Digital Pin 8.
- Pins 7-10 (D0-D3): Leave unconnected (since we are using 4-bit mode).
- Pin 11 (D4): Connect to Arduino Digital Pin 9.
- Pin 12 (D5): Connect to Arduino Digital Pin 10.
- Pin 13 (D6): Connect to Arduino Digital Pin 11.
- Pin 14 (D7): Connect to Arduino Digital Pin 12.
- Pin 15 (LED+ - Backlight Anode): Connect through a 220 Ohm resistor to the breadboard's positive (+) power rail.
- Pin 16 (LED- - Backlight Cathode): Connect to the breadboard's negative (-) ground rail.
5. Connect the Potentiometer (for Speed Control):
- Outer Pin 1 (Terminal 1): Connect to the breadboard's positive (+) power rail.
- Outer Pin 2 (Terminal 2): Connect to the breadboard's negative (-) ground rail.
- Middle Pin (Wiper): Connect to Arduino Analog Pin A0.



Code Explanation

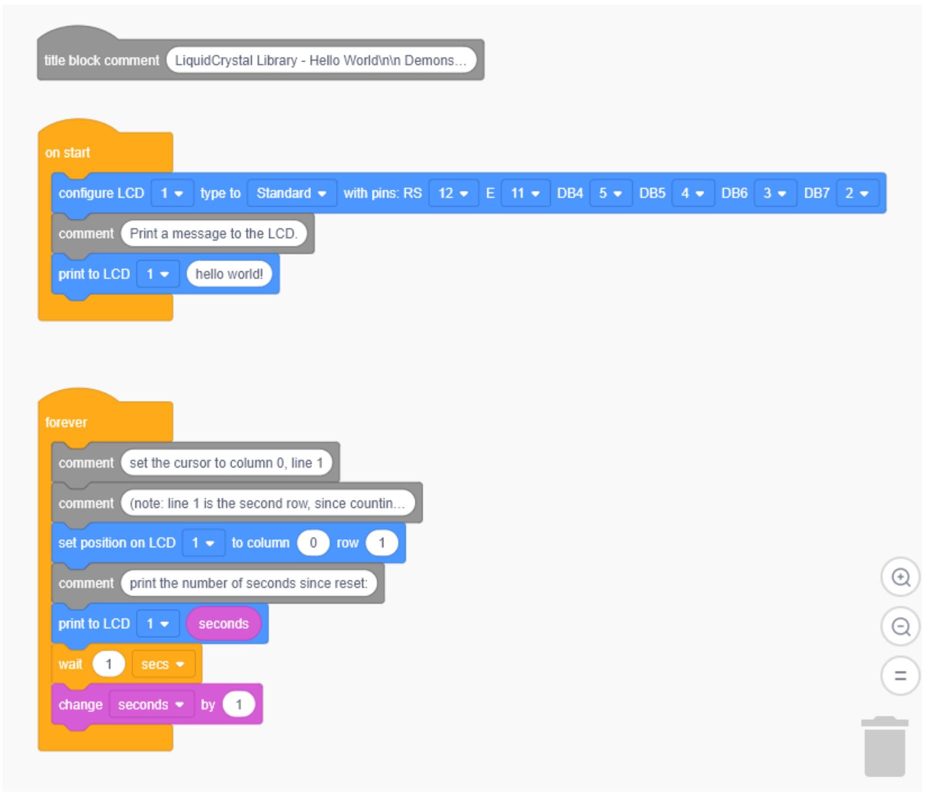
The provided Arduino C++ based code will control the LCD display and read input from the potentiometer.

```
// C++ code
//
/*
  LiquidCrystal Library - Hello World

   Demonstrates the use of a 16x2 LCD display.
  The LiquidCrystal library works with all LCD
  displays that are compatible with the  Hitachi
  HD44780 driver. There are many of them out
  there, and you  can usually tell them by the
  16-pin interface.

  This sketch prints "Hello World!" to the LCD
  and shows the time.

  The circuit:
  * LCD RS pin to digital pin 12
  * LCD Enable pin to digital pin 11
  * LCD D4 pin to digital pin 5
```

```
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)

 Library originally added 18 Apr 2008  by David
 A. Mellis
 library modified 5 Jul 2009  by Limor Fried
 (http://www.ladyada.net)
 example added 9 Jul 2009  by Tom Igoe
 modified 22 Nov 2010  by Tom Igoe

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/LiquidCrystal
*/

#include <LiquidCrystal.h>

int seconds = 0;

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);

void setup()
{
 lcd_1.begin(16, 2); // Set up the number of columns and rows on the LCD.
```

```
  // Print a message to the LCD.
  lcd_1.print("hello world!");
}

void loop()
{
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting
  // begins with 0):
  lcd_1.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd_1.print(seconds);
  delay(1000); // Wait for 1000 millisecond(s)
  seconds += 1;
}
```

Block Code Overview:



```
title block comment  LiquidCrystal Library - Hello World\n\n Demons...

on start
  configure LCD 1 type to Standard with pins: RS 12 E 11 DB4 5 DB5 4 DB6 3 DB7 2
  comment  Print a message to the LCD.
  print to LCD 1  hello world!

forever
  comment  set the cursor to column 0, line 1
  comment  (note: line 1 is the second row, since countin...
  set position on LCD 1 to column 0 row 1
  comment  print the number of seconds since reset:
  print to LCD 1  seconds
  wait 1 secs
  change seconds by 1
```

Expected Outcome/Operation

Upon successful simulation in TinkerCAD:

• The LCD 16x2 will first display "Hello World!" on its top line.

• The second line of the LCD will display "Count: 0" initially, which will then start incrementing after the first interval.

• Rotating the potentiometer's knob will directly influence the rate at which the counter increments on the LCD. Turning the potentiometer will adjust the interval from 100ms (fastest) to 5000ms (slowest).

• The Serial Monitor in TinkerCAD will display the counter value each time it increments, allowing you to verify the program's logic.

Troubleshooting Tips

If your circuit does not work as expected during simulation, consider the following common mistakes:

• Wiring Errors: Carefully double-check all your wire connections against the circuit assembly instructions. Even a single misplaced wire can prevent the circuit from functioning. Ensure wires are pushed "all the way" into the breadboard holes.

• Polarity: Verify that components with polarity (like the LCD's backlight LED) are connected in the correct direction (e.g., LED anode to positive, cathode to negative via a resistor).

• Component Values: Ensure resistors are of the correct value (e.g., 220 Ohm for LCD backlight).

• LCD Contrast (VEE): If the LCD display is blank or shows solid blocks, adjust the potentiometer connected to its VEE pin, or ensure Pin 3 (VEE) is correctly wired.

• Arduino Pin Mapping: Confirm that the pin numbers specified in the LiquidCrystal lcd(...) line in your code precisely match the Arduino digital pins you've connected to the LCD's RS, E, D4, D5, D6, D7 pins.

• Code Syntax: While TinkerCAD's debugger can help, minor typos or missing semicolons in C++ code can cause issues.

• Short Circuits: Be vigilant for any accidental connections between power (5V) and ground (GND) or between other component leads that shouldn't be connected, as this can prevent the circuit from working or even cause damage. TinkerCAD might show warnings if a short circuit is detected.