

**MICROCONTROLLERS FOR MECHATRONICS – MECA442**

Experiment 9: Design of a calculator using arduino

**Jana Kabrit1, Oussama Al Ahmad Matar1**

**1** Rafik Hariri University, College of Engineering, Department of Mechanical and Mechatronics

*Abstract*

*The aim of this experiment is to design and simulate a calculator using an Arduino Uno board. A keypad will be used as the only source for input and an LCD screen for the output. The input consists of numbers and standard mathematical operators, and the Arduino would perform the specified operation printing out the result. The experiment was successful, and the desired results were obtained.*

**Keywords:** Arduino Uno, Calculator, Math Operations, Keypad, LCD Screen, TinkerCAD.

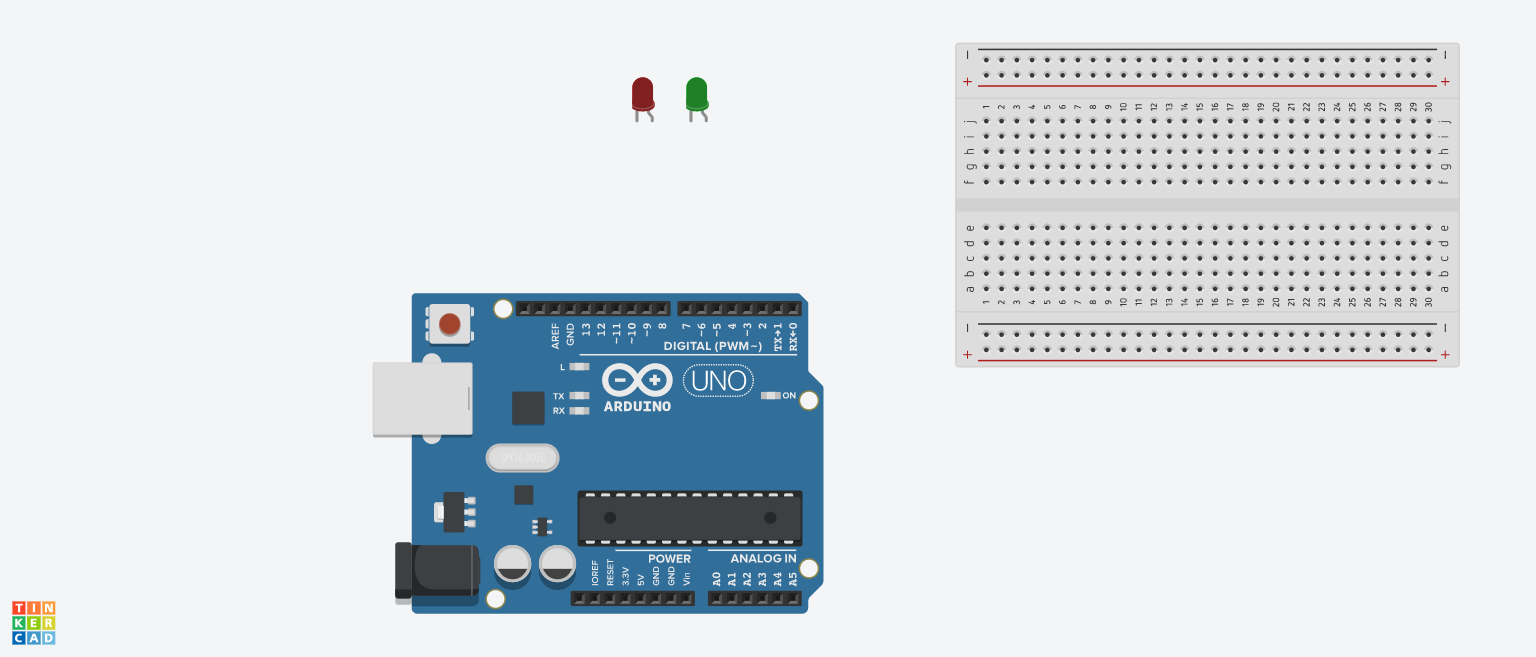
1. INTRODUCTION

In this experiment, a calculator will be designed using an Arduino board, keypad, and LCD screen. A 4x4 keypad will act as the input, allowing digits from 0 to 9, the standard mathematical operators addition, subtraction, multiplication, and divisions, as well as a button for equal and clear. The user will be able to perform operations between two numbers only using one operator, a certain number is written, followed by an operator, then a second number. The size of the numbers is tied to the bit size of the Arduino Uno, and since the numbers are coded as long integers, they’re extended and can reach the theoretical range of 232 numbers, which equates to -2,147,483,648 to 2,147,483,647. However, such numbers are too big to be displayed on the LCD screen being used. After inputting the numbers and an operator, the user can press the equal key for the result, and the clear key to start over and reset at any time during the process.

1. **MATERIALS AND METHODS**
   1. **Materials**
      1. **Simulated Electronics Components**

* Arduino Uno

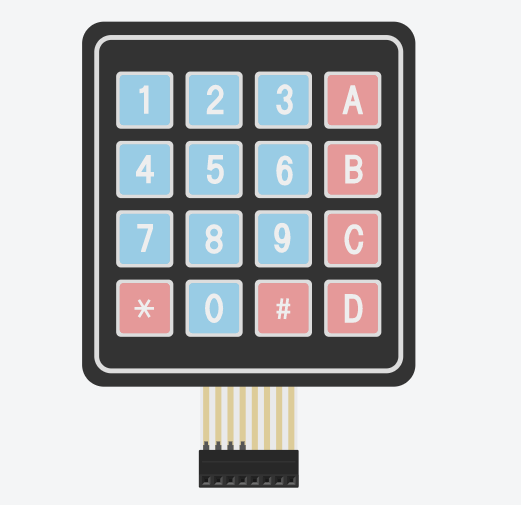
The Arduino Uno shown in Figure 1 is a microcontroller board based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.[1]



**FIGURE 1:** ARDUINO UNO BOARD

In this experiment, the Arduino is required so that it can be programmed to input the numbers and operators from the keypad and output them on the LCD screen, then perform the operation needed just as a calculator would.

* Keypad:



**FIGURE 2:** 4X4 KEYPAD

A keypad is a set of buttons which arranged in rows and columns. A 3X4 keypad has 4 rows and 3 columns, and a 4X4 keypad has 4 rows and 4 columns. Beneath each key is a membrane switch. Each switch in a row is connected to the other switches in the row by a conductive trace underneath the pad. Each switch in a column is connected the same way – one side of the switch is connected to all of the other switches in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 8 pins on a 4X4 keypad. Pressing a button closes the switch between a column and a row trace, allowing current to flow between a column pin and a row pin. [2]

In this experiment, the number keys will be used from a 4X4 keypad to be able to input the numbers on the LCD, as well as letters A, B, C, and D as the operators for addition, subtraction, multiplication, and division respectively. The asterisk (\*) symbol will be used as clear, and the hashtag (#) symbol as equal.

* LCD Screen:

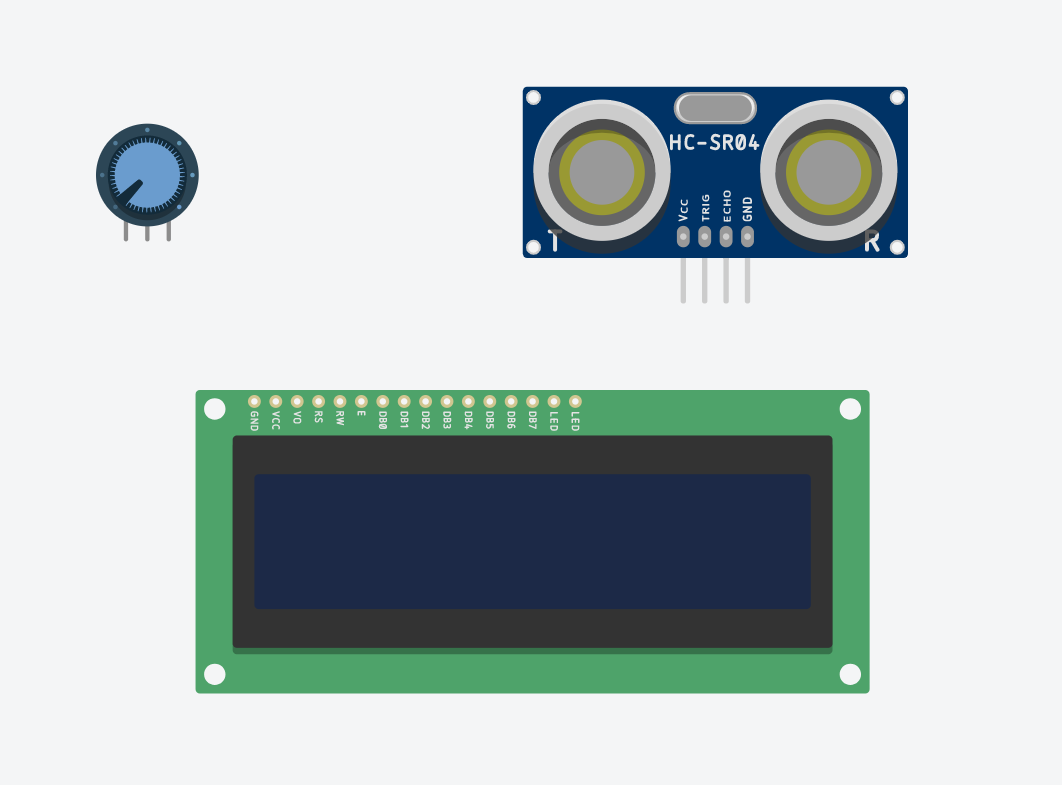
These LCDs are available in many different sizes (16×2 1602, 20×4 2004, 16×1 etc.), but they all use the same HD44780 parallel interface LCD controller chip from Hitachi. This means you can easily swap them. The LCD has 16 connection pins, numbered 1-16 from left to right. If the display does not include a resistor, you will need to add one between 5 V and pin 15. It should be safe to use a 220Ω resistor, but this value might make your display a bit dim. In this experiment, we used a potentiometer to get the best brightness. The maximum current rating of the backlight can be checked from the and used this to select an appropriate resistor value.[3]



**FIGURE 3:** LIQUID CRYSTAL DISPLAY

* Potentiometer:

The potentiometer, commonly referred to as a “pot”, is a three-terminal mechanically operated rotary analogue device which can be found and used in a large variety of electrical and electronic circuits. They are passive devices, meaning they do not require a power supply or additional circuitry in order to perform their basic linear or rotary position function.[4] In this experiment, we will use the potentiometer to get the best brightness of the LCD screen available.



**FIGURE 4:** POTENTIOMETER

* + 1. **Code Components**
* const: A variable qualifier that modifies the behavior of the variable, making a variable "read-only" and constant.
* long int: Long variables are extended size variables for number storage, and store 32 bits (4 bytes) instead of the usual 16 bits (2 bytes).
* unsigned int: Same as int in that it stores a 2-byte value. Instead of storing negative numbers however they only store positive values, yielding a useful range of 0 to 65,535.
* void setup: The function is called whenever the program starts. It is used to initialize variables, pin modes, etc. It will only run once after each powerup or reset of the Arduino board.
* pinMode(): A function used to configure a specific pin to behave either as an input or an output, usually in the void setup.
* void loop: The function that holds the code inside and runs over and over as long as the board is turned on.
* millis(): A command that returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.
* LiquidCrystal lcd: Provides a set of endpoints to manage Arduino IoT Cloud Devices, Things, Properties and Timeseries. This API can be called just with any HTTP Client, or using one of these clients
* lcd.setCursor(): Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display
* lcd.print(): Displays what is in the argument on the LCD Screen
* lcd.clear(): Clears the LCD screen and positions the cursor in the upper-left corner.
  1. **Methods**
     1. **Code Description**

Once the Arduino Uno turns on, it sets up the pins for the LCD display (in our case pins A5, 1, 2, 3, 4, and 5 as the RS, enable, D4, D5, D6, and D7 respectively) to be able to print the output on it. Then sets up the pins for the keypad as rows and columns (in our case pins 9, 8, 7, and 6 for columns 1 to 4, and pins 10, 11, 12, and 13 for rows 1 to 4). It also assigns the integer values for addition as 10, subtraction as 11, multiplication as 12, division as 13, equal as 14, and clear as 15 since the keypad has 16 keys. The Booleans Check, Sign, and Done are initially set to false. The first and second numbers are initially set to zero.

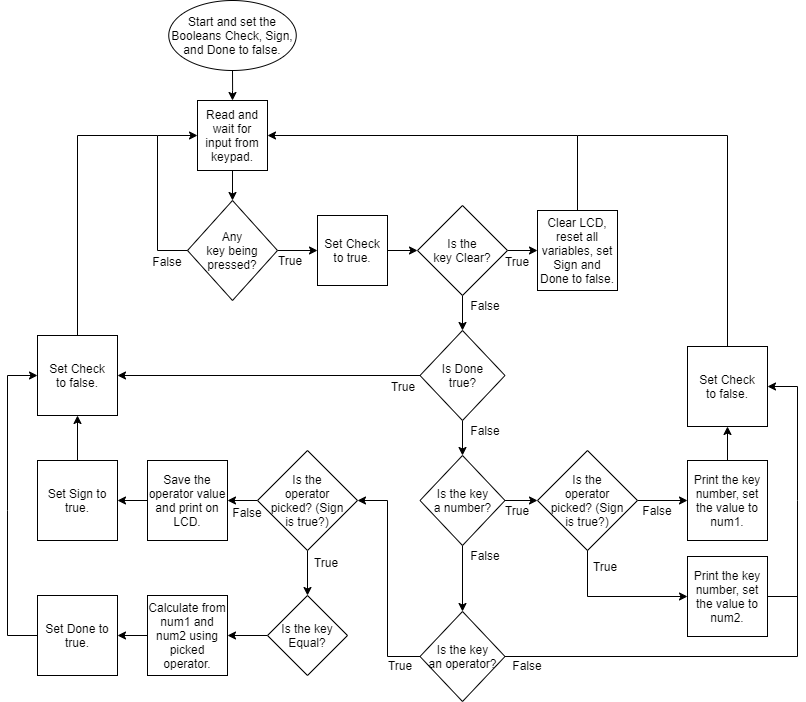
The rows functions are responsible for detecting any input from the keypad with respect to the columns, and a debounce function is used to read from each key instead of using digitalRead to avoid pressing the buttons multiple times, with a debounce interval of 100 milliseconds. The check Boolean is responsible for checking whether a button has been pressed to avoid looping the value of the key in the code. The sign Boolean checks whether an operator has been chosen yet since the user can only input two numbers, a number before and another after the operator. The done Boolean is responsible for checking if the user calculated an operating to avoid further inputs other than clear.

In the void loop, when a user presses a key, the key’s value is first compared to the value of the clear key which is set to 15, and clears the LCD and resets all variables if it was equal. Otherwise, if done is set to false, the key’s value is compared to 9. If it’s less than or equal to 9 then a number is being pressed, to know if it’s the first number or the second number, the sign Boolean comes in place, since if the user hasn’t picked an operator then it’s the first number, otherwise, it would be the second that comes after the operator and each key press is printed individually on the LCD. A function is used to properly store the value of a number which multiplies the old value by 10 and adds the new key value to it. Once an operator has been picked, the sign Boolean is set to true to indicate that any further number keys pressed are for the second number.

For keys with values greater than 9 which are operators and equal. If the sign Boolean is false which indicates an operator hasn’t been picked, the operator is then printed on the LCD and its value is saved for use later. If the sign Boolean is true, then if the key pressed is equal, the Arduino would calculate the result of the first and second number according to the value of the operator picked earlier. Finally, the check Boolean is set to false so another key can be pressed without indefinitely looping in the void loop. The values of each key are shown in Table 1 and a flowchart is shown describing the code process in Figure 5.

**TABLE 1:** KEY FUNCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Function** | **Key** | **Function** |
| 0 | Number 0 | 8 | Number 8 |
| 1 | Number 1 | 9 | Number 9 |
| 2 | Number 2 | A (10) | Add |
| 3 | Number 3 | B (11) | Subtract |
| 4 | Number 4 | C (12) | Multiply |
| 5 | Number 5 | D (13) | Divide |
| 6 | Number 6 | # (14) | Equal |
| 7 | Number 7 | \* (15) | Clear |



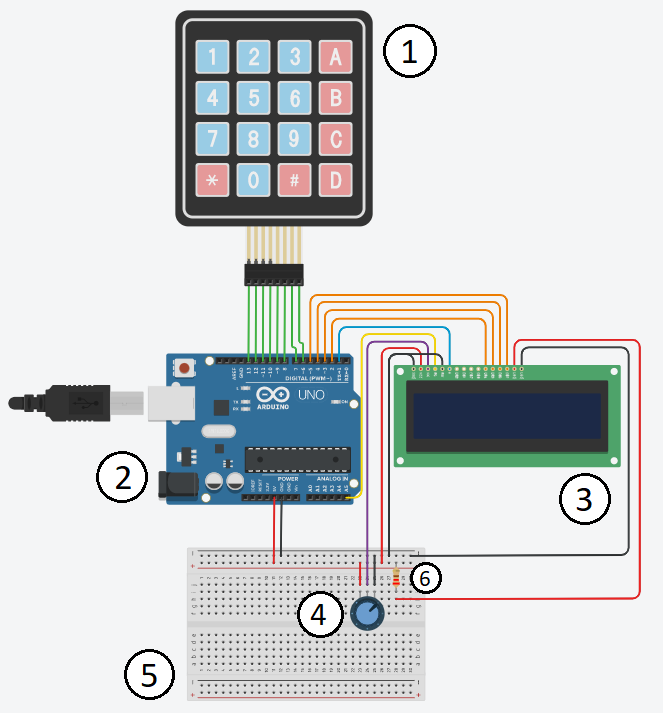
**FIGURE 5:** FLOWCHART

* + 1. **Connections on Tinker CAD**

1. First, connect the 5V and ground pins of the Arduino Uno to the breadboard power and ground (red and black wires).
2. For the LCD, connect the data pins D4, D5, D6, and D7 to the Arduino at pins 2, 3, 4, and 5 respectively to be able to print on the display (orange wires). The ground, R/W, and cathode pins are connected to the ground pin of the breadboard, and both the VCC and anode of the LCD are connected to the 5V of the breadboard, however, the anode is first connected to a 220Ω resistor to avoid damaging the screen.
3. The RS pin of the LCD is connected to pin A5 (yellow wire) but will still be used as a digital pin, and the Enable to pin 1 (blue wire), and finally V0 is connected to a potentiometer (purple wire) which is connected to the 5V and ground of the breadboard.
4. Finally, for the keypad, the row pins 1, 2, 3, and 4 are connected to the Arduino pins 13, 12, 11, and 10 respectively. The column pins 1, 2, 3, and 4 are connected to 9, 8, 7, and 6 of the Arduino (green wires).
5. **RESULTS AND DISCUSSION**
   1. **Simulation**

We implemented the circuit on Tinker CAD, adding the code to the Arduino Simulation, obtaining the circuit shown in Figure 6.

**TABLE 2:** LEGEND



**FIGURE 6:** CIRCUIT SCHEMATIC ON TINKER CAD

|  |  |
| --- | --- |
| NUMBER | COMPONENT |
| 1 | 4x4 Keypad |
| 2 | Arduino Uno Board |
| 3 | 16x2 Liquid Crystal Display |
| 4 | Potentiometer |
| 5 | Breadboard |
| 6 | 220Ω Resistor |

* 1. **Analysis**

After running the simulation, upon pressing any number key, the pressed number is displayed on the LCD screen, and pressing more number keys adds more digits to the pre-written number and increases it just like a traditional calculator would, then saves its value as the first number since it was written before choosing an operator. Pressing an operator key printed the operator on the screen and saved its value for later use. Then, pressing more number keys after the picked operator had the same process as before, except the value was stored in the second number which is after the operator. Pressing the equal key resulted in an equal sign on a new line with the correct answer printed on the LCD screen whether it was addition, subtraction, multiplication, and division. The clear button cleared the LCD and reset all values at any instant during the process. Moreover, trying to press multiple operators does not work nor are printed on the LCD, only the first operator pressed is the one that is taken and printed. Also, pressing any key after the resulted answer has been printed does not work, except the clear key to start a new operation. A video of the simulation can be shown through [this link](https://drive.google.com/file/d/16VI-GuCwSjTHV927S3YkT3aGamX4Z_Oq/view?usp=sharing), and the shared TinkerCAD simulation can be accessed on [this link](https://www.tinkercad.com/things/h3uLVvdlqbg).

1. **CONCLUSION**

The objective of this experiment is to design a calculator that performs basic mathematical operations using two numbers. The two numbers can have a value range of 232 and can result in a decimal number when needed in division. The equal key gives the result of the operation on a new line and the clear key resets all the variables and clears the LCD display at any instant in the process. The experiment was successful, and the simulation behaved exactly like we programmed it to.

**REFERENCES**

[1] Pololu Robotics & Electronics. *Arduino Uno*, 2020

https://www.pololu.com/product/2191#:~:text=Overview,header%2C%20and%20a%20reset%20button.

[2] Circuit Basics, *How to Set Up a Keypad On An Arduino*, 2020

https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino

[3] Maker Guides, *How to use a 16×2 character LCD with Arduino*, 2020

https://www.makerguides.com/character-lcd-arduino-tutorial/

[4] Electronics Tutorials, *Potentiometers*, 2020

https://www.electronics-tutorials.ws/resistor/potentiometer.html

[5] Autodesk Tinkercad  
 https://www.tinkercad.com/

**Appendix**

* Arduino Code:

#include <LiquidCrystal.h>

LiquidCrystal lcd(A5, 1, 2, 3, 4, 5);

const int c1 = 9, c2 = 8, c3 = 7, c4 = 6;

const int r1 = 13, r2 = 12, r3 = 11, r4 = 10;

int add = 10, subtract = 11, multiply = 12, divide = 13;

int equal = 14, clear = 15;

unsigned long previousDebounceTime = 0;

unsigned long debounceDelay = 100;

long int num1 = 0, num2 = 0, operand, placeholder;

double result;

bool check = false, sign = false, done = false;

void setup() {

lcd.begin(16,2);

lcd.setCursor(0, 0);

lcd.print("Arduino Uno");

lcd.setCursor(0,1);

lcd.print("Calculator!");

delay(1000);

lcd.clear();

pinMode(c1, INPUT\_PULLUP);

pinMode(c2, INPUT\_PULLUP);

pinMode(c3, INPUT\_PULLUP);

pinMode(c4, INPUT\_PULLUP);

pinMode(r1, OUTPUT);

pinMode(r2, OUTPUT);

pinMode(r3, OUTPUT);

pinMode(r4, OUTPUT);

}

void loop() {

row1();

row2();

row3();

row4();

if(check == true){

if(placeholder == clear){

lcd.clear();

num1 = 0;

num2 = 0;

operand = 0;

placeholder = -1;

sign = false;

done = false;

}

}

if(check == true && done == false){

if(placeholder <= 9 && placeholder != -1){

if(sign == false){

lcd.print(placeholder);

num1 = (num1 \* 10) + placeholder;

}

if(sign == true){

lcd.print(placeholder);

num2 = (num2 \* 10) + placeholder;

}

}

if(placeholder > 9){

if(sign == false){

operand = placeholder;

if(operand == add)

lcd.print("+");

else if(operand == subtract)

lcd.print("-");

else if(operand == multiply)

lcd.print("x");

else if(operand == divide)

lcd.print("/");

sign = true;

}

if(sign == true){

if(placeholder == equal){

if(operand == add)

result = num1 + num2;

else if(operand == subtract)

result = num1 - num2;

else if(operand == multiply)

result = num1 \* num2;

else if(operand == divide)

result = (double)num1 / (double)num2;

done = true;

lcd.setCursor(0,1);

lcd.print("=");

lcd.print(result);

}

}

}

check = false;

}

}

void row1() {

digitalWrite(r1, LOW);

digitalWrite(r2, HIGH);

digitalWrite(r3, HIGH);

digitalWrite(r4, HIGH);

if (debounce(c1) == LOW) {

placeholder = 1;

}

else if (debounce(c2) == LOW) {

placeholder = 2;

}

else if (debounce(c3) == LOW) {

placeholder = 3;

}

else if (debounce(c4) == LOW) {

placeholder = 10;

}

}

void row2() {

digitalWrite(r1, HIGH);

digitalWrite(r2, LOW);

digitalWrite(r3, HIGH);

digitalWrite(r4, HIGH);

if (debounce(c1) == LOW) {

placeholder = 4;

}

else if (debounce(c2) == LOW) {

placeholder = 5;

}

else if (debounce(c3) == LOW) {

placeholder = 6;

}

else if (debounce(c4) == LOW) {

placeholder = 11;

}

}

void row3() {

digitalWrite(r1, HIGH);

digitalWrite(r2, HIGH);

digitalWrite(r3, LOW);

digitalWrite(r4, HIGH);

if (debounce(c1) == LOW) {

placeholder = 7;

}

else if (debounce(c2) == LOW) {

placeholder = 8;

}

else if (debounce(c3) == LOW) {

placeholder = 9;

}

else if (debounce(c4) == LOW) {

placeholder = 12;

}

}

void row4() {

digitalWrite(r1, HIGH);

digitalWrite(r2, HIGH);

digitalWrite(r3, HIGH);

digitalWrite(r4, LOW);

if (debounce(c1) == LOW) {

placeholder = 15;

}

else if (debounce(c2) == LOW) {

placeholder = 0;

}

if (debounce(c3) == LOW) {

placeholder = 14;

}

else if (debounce(c4) == LOW) {

placeholder = 13;

}

}

int debounce(int buttonPin){

int reading = digitalRead(buttonPin);

if (reading == LOW) {

if ((millis() - previousDebounceTime) > debounceDelay) {

previousDebounceTime = millis();

check = true;

return LOW;

}

}

return HIGH;

}