Arduino Digital Calculator

Harry Liu

Table of Content

Description

General Description

This circuit is a system that has some of the basic functions of a working calculator. It allows the user to enter two single digit numbers, and choose between one of the four operation choices (+, -, *, /).

Order of Input

Numbers and signs need to be inputted in order, system will not allow inputs that are in the wrong order. (The system won't do anything)

Reset switch can be pressed at any time to clear everything and restore the calculator to default condition. (Even at the end of the calculation if the user wants to make more calculations)

> 💡 Order of Input: number1 → operation sign → number2 → equal sign. (Reset at any time)

After the equal sign is pressed, the system will then calculate the answer and display it on two 7-segment displays (2 digits), and all decimal values will be truncated.

For example:

- 1 will be displayed as 01
- 81 will be displayed as 81
- 0 will be displayed as 00
- 2.25 will be displayed as 02
- 0.56 will be displayed as 00

Symbol Representation

It is important to identify what each symbol/LED represent.

Keypad

LEDs

▼ Addition

Symbol A on the keypad

▼ Subtraction

Symbol B on the keypad

▼ Multiplication

Symbol C on the keypad

▼ Division

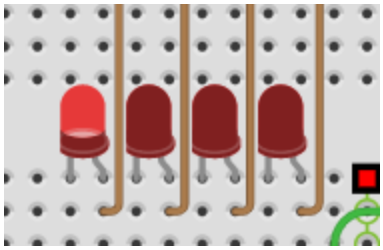
Symbol D on the keypad

▼ Equal Sign

symbol on the keypad

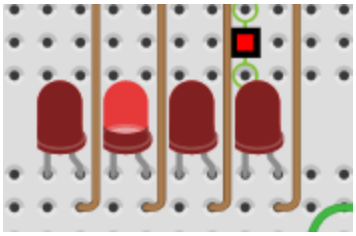
▼ Addition

First LED of the four that are grouped together



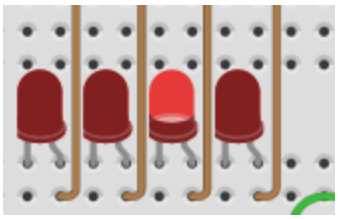
▼ Subtraction

Second LED of the four that are grouped together



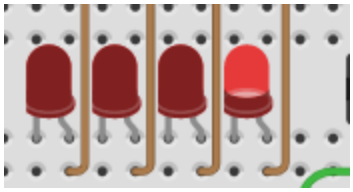
▼ Multiplication

Third LED of the four that are grouped together



▼ Division

Fourth LED of the four that are grouped together



▼ Reset Switch

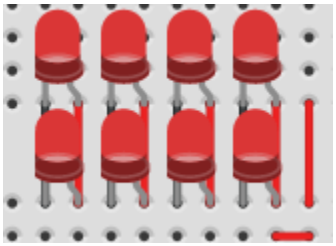
* symbol on the keypad

▼ Numbers (0-9)

Numbers through 0-9 on the keypad, already labelled.

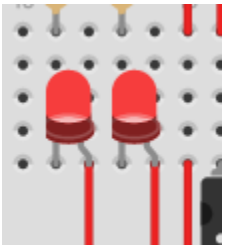
▼ Equal Sign

The 8 LEDs grouped together



▼ Negative Sign

The 2 LEDs grouped together



▼ Input Numbers & Output Numbers

Input numbers are displayed on the 2 7-segment displays placed on the left breadboard.

Output numbers are displayed on the 2 7-segment displays placed on the right breadboard.

In-depth Description (Optional)

Number/Sign Changes

Reminder:

> 💡 Order of Input: number1 → operation sign → number2 → equal sign. (Reset at any time)

As mentioned earlier, the program will not accept any inputs that are in the wrong order. For example, it will not accept operation signs to be entered before number1 is entered, or after number2 is entered. Pressing the keys in the wrong order will not do anything.

However, the program does allow changes to be made, without pressing the reset switch. It allows the users to change their inputs (both numbers and signs) when the input index is at that particular point. For example, number1 can be changed however many times the user wants to as long as an operation sign is not pressed (as soon as any operation sign is pressed the input index would jump to the next index for signs). Same concept applies to operation sign input and number2 input as well. As long as number2 is not entered, the user can change the operation sign however many times they intend. And as long as the equal sign is not pressed, number2 can be freely changed as well.

However, if for example, an operation sign is already entered and now the user wants to change number1, the system would not allow this so the only way would be to reset the calculator.

LED/Display Lighting Pattern

At start-up, without any manual inputs, the four 7-segment displays would output 0, and all the LEDs (operation sign, equal sign, negative sign) would be off.

Whenever a number is entered/calculated and now ready to be displayed, its corresponding 7-segment display would change based on the number provided. (will not change if the number is 0)

The operation sign LED will only light up after the first time an operator is entered, and it will stay on throughout the entire calculation process. It will only turn off when the reset switch is pressed.

The equal sign LEDs will only turn on when the equal sign is pressed, in the correct order (pressing it in the wrong order will not do anything). It will stay on until the reset switch is pressed.

The negative sign LEDs will only light up if the answer is a negative number, which would only occur when number1 < number2 in subtraction operation. It will turn on with the equal sign (if necessary) and stay on until reset switch is pressed.

Arduino 1 Functions & Pseudocode

Arduino 1 is responsible for displaying the input numbers and communicating with Arduino 3 (tell Arduino 3 which operation sign LED to light up).

Pseudocode:

```
Initialize variables;
setup;

loop {
  get keypadInput;
  if (keypadInput != blank) {

    //number input
    if (keypadInput == a number) {

      if (index == 0 || index == 2) {  //adjust index
        index++;
      }

      if (index == 1) {   //num1
        convertToBinary;
        displayNumber;
      }
      if (index == 3) {   //num2
        convertToBinary;
        displayNumber;
      }
    }

    //operation sign
    if (keypadInput == an operation sign) {
      if (index == 1) {  //adjust index
        index++;
      }
      if (index == 2) {    //operation sign
        send signal to Arduino 3;
      }
    }

    //reset
    if (keypadInput == resetSwitch) {
      reset everything;
    }
  }
}
```

Arduino 2 Functions & Pseudocode

Arduino 2 is responsible for doing calculations given the inputs, displaying the answers, and communicating with Arduino 3 to indicate when to light up operation sign LEDs, equal sign LEDs, and negative sign LEDs

Pseudocode:

```
Initialize variables;
setup;

loop {
  get keypadInput;
  if (keypadInput != blank) {

    //number input
    if (keypadInput == a number) {

      if (index == 0 || index == 2) {  //adjust index
        index++;
      }

      if (index == 1) {   //num1
        store input;
      }
      if (index == 3) {   //num2
        store input;
      }
    }

    //operation sign
    if (keypadInput == an operation sign) {
      if (index == 1) {  //adjust index
        index++;
      }
      if (index == 2) {    //operation sign
        store operation sign;
      }
    }

    //equal sign pressed, only executes when all 3 inputs are entered
    if (keypadInput == equal sign && index == 3) {

      communicate with Arduino 3, turn on equal sign LEDs;
      calculate and store;
      communicate with Arduino 3 when necessary for negative sign LEDs;

      convertToBinary;
      displayNumber;
    }

    //reset
    if (keypadInput == resetSwitch) {
```

```
      reset everything;
    }
  }
}
```

Arduino 3 Functions & Pseudocode

The Arduino 3 is mainly responsible for taking input commands from Arduino 1 and 2, and executing these commands in the form of turning on LEDs.

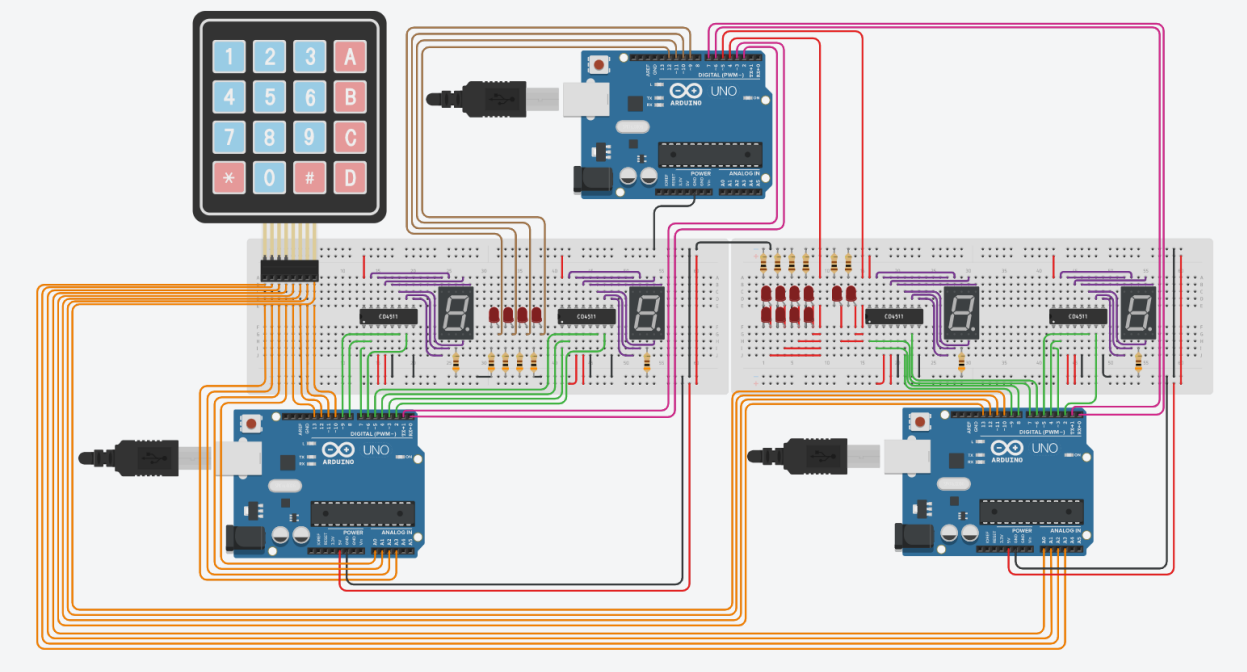
It takes commands from Arduino 1 telling it which operation sign LED to turn on. And commands from Arduino 2 telling it when to turn on the operation signs, equal sign, and negative sign.

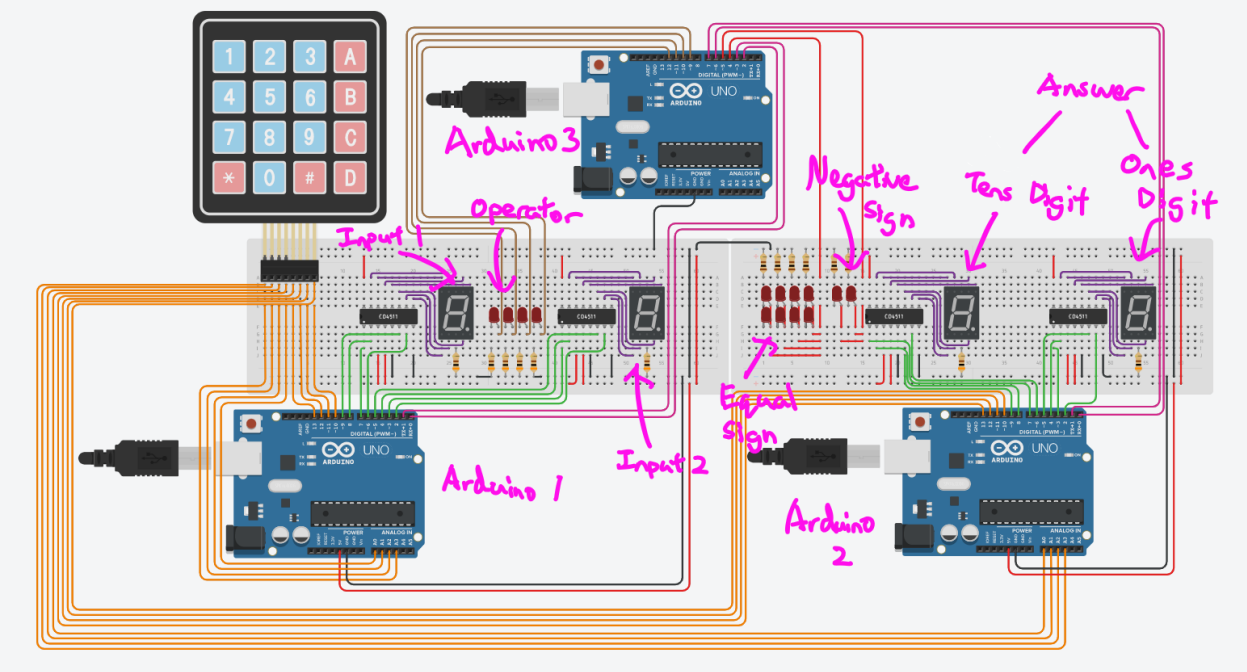
Pseudocode:

```
Initialize variables;
setup;

loop {
  if (!operatorEntered) {  //if no operator has been entered yet
    delay(10);   //accommodate for the possibilities of initial start-up issues.
  }

  //only allow operator LEDs to light up when at least one
  //operator has been entered by the user
  if (Operator has been entered && !operatorEntered) {
    operatorEntered = true;
  }

  //turn on operation sign LEDs
  if (operatorEntered) {
    light up specified operation sign LED;
    turn other three LEDs off;
  }

  //turn on equal sign LEDs
  if (equal sign is pressed) {
    turn on equal sign LEDs;
  }

  //turn on negative sign LEDs
  if (negative sign is required) {
    turn on negative sign LEDs;
  }

  //reset
  //check if operatorEntered is true to avoid unecessary resets
  //especially the ones that will cause problems in the system.
```

```
  if (reset key pressed && operatorEntered) {
    reset everything;
  }

}
```

Circuit Screen Shot





Code

Arduino 1

```
/*
Name: Harry Liu
*/

/*
  Code for Arduino 1. Mainly to control the displays of the user input
Also transfers commands to Arduino 3 for operation sign display
*/

#include <Keypad.h>


//initialize variables

//For the keypad
const byte ROWS = 4;
const byte COLS = 4;
const byte C1 = 13;
const byte C2 = 12;
const byte C3 = 11;
const byte C4 = 10;
const byte R1 = A3;
const byte R2 = A2;
```

```
const byte R3 = A1;
const byte R4 = A0;
char keys[ROWS][COLS] = { {'1','2','3','A'},
                          {'4','5','6','B'},
                          {'7','8','9','C'},
                          {'*','0','#','D'}
                        };
byte rowPins[ROWS] = {A3, A2, A1, A0};  //row pin numbers
byte colPins[COLS] = {13, 12, 11, 10};  //col pin numbers
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

//for the 7-segment decoders
const byte firstNum1 = 6;
const byte firstNum2 = 7;
const byte firstNum3 = 8;
const byte firstNum4 = 9;
const byte secondNum1 = 2;
const byte secondNum2 = 3;
const byte secondNum3 = 4;
const byte secondNum4 = 5;

//sign communication to Arduino 3
const byte wire0 = 0;
const byte wire1 = 1;

//binary arrays, store binary numbers
byte binaryInputLeft[4];
byte binaryInputRight[4];

byte index = 0; //index pointer used to keep track of input order

//initial setup
void setup() {
  pinMode(C1, INPUT);
  pinMode(C2, INPUT);
  pinMode(C3, INPUT);
  pinMode(C4, INPUT);
  pinMode(R1, INPUT);
  pinMode(R2, INPUT);
  pinMode(R3, INPUT);
  pinMode(R4, INPUT);

  pinMode(firstNum1, OUTPUT);
  pinMode(firstNum2, OUTPUT);
  pinMode(firstNum3, OUTPUT);
  pinMode(firstNum4, OUTPUT);
  pinMode(secondNum1, OUTPUT);
  pinMode(secondNum2, OUTPUT);
  pinMode(secondNum3, OUTPUT);
  pinMode(secondNum4, OUTPUT);

  pinMode(wire0, OUTPUT);
  pinMode(wire1, OUTPUT);
}
```

```
void loop() {

  char key = keypad.getKey(); //get an input from keypad

  if (key != NO_KEY) {  //if a key is pressed
    //if key pressed is a number, display
    if (key == '1' || key == '2' || key == '3' ||
        key == '4' || key == '5' || key == '6' ||
        key == '7' || key == '8' || key == '9' || key == '0') {

      //if index is pressed at the very beginning
      //or after an operation sign, index++
      if (index == 0 || index == 2) {
        index++;
      }

      byte input = (byte)key - 48;  //gets the input as a byte

      //index == 1, meaning the first number, only change the first display
      if (index == 1) {
        binaryArrayReset(binaryInputLeft);    //reset binary array
        convertToBinary(input, binaryInputLeft);//decimal to binary

        turnOff(firstNum1);           //turn everything off
        display(firstNum1, binaryInputLeft);  //display the number
      }

      //index == 3, meaning the second number, only change the second display
      if (index == 3) {
        binaryArrayReset(binaryInputRight);   //reset binary array
        convertToBinary(input, binaryInputRight);//decimal to binary

        turnOff(secondNum1);          //turn everything off
        display(secondNum1, binaryInputRight);  //display the number
      }
    }

    //If an operation sign is entered, communicate with Arduino 3
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') {

      //changes index only when it is the first time coming from
      //an integer input (doesn't change for operation changes)
      if (index == 1) {
        index++;
      }

      //only enters when index == 2
      if (index == 2) {
        switch (key) {
          case 'A': //A = add, both wires off
            digitalWrite(wire0, LOW);
            digitalWrite(wire1, LOW);
```

```
            break;
          case 'B': //B = subtract, wire0 on, wire1 off
            digitalWrite(wire0, HIGH);
            digitalWrite(wire1, LOW);
            break;
          case 'C': //C = multiply, wire0 off, wire1 on
            digitalWrite(wire0, LOW);
            digitalWrite(wire1, HIGH);
            break;
          case 'D': //D = divide, both wires on
            digitalWrite(wire0, HIGH);
            digitalWrite(wire1, HIGH);
            break;
        }
      }
    }

    //if reset key is pressed, reset everything to default condition
    if (key == '*') {
      turnOff(firstNum1);
      turnOff(secondNum1);
      digitalWrite(wire0, LOW);
      digitalWrite(wire1, LOW);
      binaryArrayReset(binaryInputLeft);
      binaryArrayReset(binaryInputRight);
      index = 0;
    }

  }

}


/*
  void binaryArrayReset ()
  byte binaryArray[] - This parameter is the array to reset.
               Reference is passed, same memory pointer.
  This method resets the given array (sets all values to 0)
*/
void binaryArrayReset (byte binaryArray[4]) {
  for (int i=0; i<4; i++) {
    binaryArray[i] = 0;
  }
}

/*
  void converToBinary ()
  byte tempInput - This parameter is the decimal number to be converted
  byte binaryArray[] - This parameter is the array to store the binary in.
                Reference is passed, same memory pointer.
  This method converts the binary equivalent of the given decimal number
*/
void convertToBinary (byte tempInput, byte binaryArray[4]) {
  byte count = 0;
```

```
  while (tempInput > 0) { //repeated division method
    binaryArray[count] = tempInput % 2;
    tempInput = tempInput/2;
    count++;
  }
}

/*
  void turnOff ()
  byte index - This parameter is the pin number to begin at
  This method turns a specific 7-segment display off (to 0)
*/
void turnOff (byte index) {
  for (int i=0; i<4; i++) {
    digitalWrite(index+i, LOW);
  }
}

/*
  void display ()
  byte index - This parameter is the pin number to begin at
  byte binaryArray[] - This parameter is the array to get binary number from.
                Reference is passed, same memory pointer.
  This method displays the number on the 7-segment display
*/
void display (byte index, byte binaryArray[4]) {
  for (int i=0; i<4; i++) {
    if (binaryArray[i] == 1) {
      digitalWrite(index+i, HIGH);
    }
  }
}
```

Arduino 2

```
/*
Name: Harry Liu
*/
/*
  Code for Arduino 2. Mainly used for calculation purposes, also displays
the answer on the two 7-segment displays. Decimal numbers are always
truncated(rounded down) into whole numbers.
  Communicates with Arduino 3, tells it when to start lighting up operation
LEDs, when to light up equal sign, when to light up negative sign, and
when to reset (switch everything off)
*/

#include <Keypad.h>

//initialize variables
```

```
//For the keypad
const byte ROWS = 4;
const byte COLS = 4;
const byte C1 = 13;
const byte C2 = 12;
const byte C3 = 11;
const byte C4 = 10;
const byte R1 = A3;
const byte R2 = A2;
const byte R3 = A1;
const byte R4 = A0;
char keys[ROWS][COLS] = { {'1','2','3','A'},
                          {'4','5','6','B'},
                          {'7','8','9','C'},
                          {'*','0','#','D'}
                        };
byte rowPins[ROWS] = {A3, A2, A1, A0};  //row pin numbers
byte colPins[COLS] = {13, 12, 11, 10};  //col pin numbers
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);

//for the 7-segment decoder
const byte firstNum1 = 6;
const byte firstNum2 = 7;
const byte firstNum3 = 8;
const byte firstNum4 = 9;
const byte secondNum1 = 2;
const byte secondNum2 = 3;
const byte secondNum3 = 4;
const byte secondNum4 = 5;

//sign communication to Arduino 3
const byte wire3 = 0;
const byte wire4 = 1;

//binary arrays, store binary numbers
byte binaryAnswerTens[4];
byte binaryAnswerOnes[4];

byte num1, num2; //two numbers inputted
byte answerTens, answerOnes; //answer, two digits
char sign; //operation sign

byte index = 0; //index pointer used to keep track of input order

//initial setup
void setup() {
  pinMode(C1, INPUT);
  pinMode(C2, INPUT);
  pinMode(C3, INPUT);
  pinMode(C4, INPUT);
  pinMode(R1, INPUT);
  pinMode(R2, INPUT);
  pinMode(R3, INPUT);
```

```
  pinMode(R4, INPUT);

  pinMode(firstNum1, OUTPUT);
  pinMode(firstNum2, OUTPUT);
  pinMode(firstNum3, OUTPUT);
  pinMode(firstNum4, OUTPUT);
  pinMode(secondNum1, OUTPUT);
  pinMode(secondNum2, OUTPUT);
  pinMode(secondNum3, OUTPUT);
  pinMode(secondNum4, OUTPUT);

  pinMode(wire3, OUTPUT);
  pinMode(wire4, OUTPUT);
}

void loop() {

  char key = keypad.getKey(); //get an input from keypad

  if (key != NO_KEY) {  //if a key is pressed
    //if key pressed is a number, store
    if (key == '1' || key == '2' || key == '3' ||
        key == '4' || key == '5' || key == '6' ||
        key == '7' || key == '8' || key == '9' || key == '0') {

      //if index is pressed at the very beginning
      //or after an operation sign, index++
      if (index == 0 || index == 2) {
        index++;
      }

      byte input = (byte)key - 48;  //gets the input as a byte

      //store the two input numbers
      if (index == 1) { //index == 1, meaning the first number
        num1 = input;
      }
      else if (index == 3) {  //index == 3, indicating the second number
        num2 = input;
      }

    }


    //If an operation sign is entered, store and communicate with Arduino 3
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') {

      //communicate with Arduino 3 for operation sign LED display
      digitalWrite(wire3, HIGH);
      digitalWrite(wire4, LOW);

      //changes index only when it is the first time coming from
      //an integer input (doesn't change for operation changes)
      if (index == 1) {
```

```
        index++;
      }

      //only enters when index == 2
      if (index == 2) {
        switch (key) {
          case 'A': //A = add
            sign = '+';
            break;
          case 'B': //B = subtract
            sign = '-';
            break;
          case 'C': //C = multiply
            sign = '*';
            break;
          case 'D': //D = divide
            sign = '/';
            break;
        }
      }
    }

    //If equal sign is pressed, calculate and display
    if (key == '#' && index == 3) { //only works when all 3 values are entered

      //communicate with Arduino 3 to ligh up equal sign LEDs
      digitalWrite(wire3, LOW);
      digitalWrite(wire4, HIGH);

      if (sign == '+') {  //add numbers
        byte sum = num1 + num2;
        answerTens = sum/10;
        answerOnes = sum-answerTens*10;
      }
      else if (sign == '-') { //subtract numbers
        if (num1 < num2) {//Tell Arduino 3 to turn on negative sign
          delay(10);  //short delay for Arduino 3 inputs...
          digitalWrite(wire3, HIGH);
          digitalWrite(wire4, HIGH);

          byte difference = num2 - num1; //calculate but the other way around
          answerTens = difference/10;
          answerOnes = difference-answerTens*10;
        }
        else {  //normal subtraction
          byte difference = num1 - num2;
          answerTens = difference/10;
          answerOnes = difference-answerTens*10;
        }
      }
      else if (sign == '*') { //multiplication
        byte product = num1 * num2;
        answerTens = product/10;
        answerOnes = product-answerTens*10;
```

```
      }
      else if (sign == '/') { //division
        if (num2 == 0) {  //cannot be divided by 0
          answerTens = 0;
          answerOnes = 0;
        }
        else {  //normal division
          byte quotient = num1 / num2;
          answerTens = quotient/10;
          answerOnes = quotient-answerTens*10;
        }
      }


      //output(display)
      convertToBinary(answerTens, binaryAnswerTens);//decimal to binary
      convertToBinary(answerOnes, binaryAnswerOnes);//decimal to binary

      display(firstNum1, binaryAnswerTens);
      display(secondNum1, binaryAnswerOnes);

    }


    //If reset switch is pressed, reset everything
    if (key == '*') {
      turnOff(firstNum1);
      turnOff(secondNum1);
      digitalWrite(wire3, LOW);
      digitalWrite(wire4, LOW);
      binaryArrayReset(binaryAnswerOnes);
      binaryArrayReset(binaryAnswerTens);
      index = 0;
      answerOnes = 0;
      answerTens = 0;
      num1 = 0;
      num2 = 0;
      sign = ' ';
    }
  }

}


/*
  void binaryArrayReset ()
  byte binaryArray[] - This parameter is the array to reset.
               Reference is passed, same memory pointer.
  This method resets the given array (sets all values to 0)
*/
void binaryArrayReset (byte binaryArray[4]) {
  for (int i=0; i<4; i++) {
    binaryArray[i] = 0;
  }
```

```
}

/*
  void converToBinary ()
  byte tempInput - This parameter is the decimal number to be converted
  byte binaryArray[] - This parameter is the array to store the binary in.
                Reference is passed, same memory pointer.
  This method converts the binary equivalent of the given decimal number
*/
void convertToBinary (byte tempInput, byte binaryArray[4]) {
  byte count = 0;
  while (tempInput > 0) { //repeated division method
    binaryArray[count] = tempInput % 2;
    tempInput = tempInput/2;
    count++;
  }
}

/*
  void turnOff ()
  byte index - This parameter is the pin number to begin at
  This method turns a specific 7-segment display off (to 0)
*/
void turnOff (int index) {
  for (int i=0; i<4; i++) {
    digitalWrite(index+i, LOW);
  }
}

/*
  void display ()
  byte index - This parameter is the pin number to begin at
  byte binaryArray[] - This parameter is the array to get binary number from.
                Reference is passed, same memory pointer.
  This method displays the number on the 7-segment display
*/
void display (byte index, byte binaryArray[4]) {
  for (int i=0; i<4; i++) {
    if (binaryArray[i] == 1) {
      digitalWrite(index+i, HIGH);
    }
  }
}
```

Arduino 3

```
/*
Name: Harry Liu
*/
```

```
/*
  Code for Arduino 3. This Arduino is used to control the LED displays
of the operation signs, equal sign, and negative sign. It receives input
from both Arduino 1 and 2. Arduino 1 tells it which operation sign to
display. Arduino 2 tells it when to start displaying the operation signs,
equal sign, negative sign, and when to reset (turn every LED off).
*/

//sign communication from Arduino 1
const byte wire0 = 3;
const byte wire1 = 2;
const byte wire3 = 7;
const byte wire4 = 6;

//output wires
const byte equalSign = 4;
const byte negativeSign = 5;
const byte addition = 9;
const byte subtraction = 10;
const byte multiplication = 11;
const byte division = 12;

bool operatorEntered = false; //used to keep every operation LED off at first

//initial setup
void setup() {
  pinMode(wire0, INPUT);
  pinMode(wire1, INPUT);
  pinMode(wire3, INPUT);
  pinMode(wire4, INPUT);

  pinMode(equalSign, OUTPUT);
  pinMode(negativeSign, OUTPUT);
  pinMode(addition, OUTPUT);
  pinMode(subtraction, OUTPUT);
  pinMode(multiplication, OUTPUT);
  pinMode(division, OUTPUT);
}

void loop() {

  //very short delay to allow system to do initial start up
  //without any chance of triggering the if statement below
  if(!operatorEntered) {
    delay(10);
  }

  //This is to make sure at the beginning every operation LED is off
  if (digitalRead(wire3) == HIGH && digitalRead(wire4) == LOW && !operatorEntered) {
    operatorEntered = true;
  }

  //only enters when flag = true, which is when at least one
  //operation sign button is pressed
```

```
  if (operatorEntered) {
    if (digitalRead(wire0) == LOW && digitalRead(wire1) == LOW) { //(+)
      digitalWrite(addition, HIGH);
      digitalWrite(subtraction, LOW);
      digitalWrite(multiplication, LOW);
      digitalWrite(division, LOW);
    }
    else if (digitalRead(wire0) == HIGH && digitalRead(wire1) == LOW){ //(-)
      digitalWrite(addition, LOW);
      digitalWrite(subtraction, HIGH);
      digitalWrite(multiplication, LOW);
      digitalWrite(division, LOW);
    }
    else if (digitalRead(wire0) == LOW && digitalRead(wire1) == HIGH){ //(*)
      digitalWrite(addition, LOW);
      digitalWrite(subtraction, LOW);
      digitalWrite(multiplication, HIGH);
      digitalWrite(division, LOW);
    }
    else if (digitalRead(wire0) == HIGH && digitalRead(wire1) == HIGH){ //(/)
      digitalWrite(addition, LOW);
      digitalWrite(subtraction, LOW);
      digitalWrite(multiplication, LOW);
      digitalWrite(division, HIGH);
    }
  }

  //equal sign
  if (digitalRead(wire3) == LOW && digitalRead(wire4) == HIGH) {
    digitalWrite(equalSign, HIGH);
  }

  //negative sign for subtraction
  if (digitalRead(wire3) == HIGH && digitalRead(wire4) == HIGH) {
    digitalWrite(negativeSign, HIGH);
  }

  //reset, check operatorEntered to avoid unecessary resets
  //that might mess up the program
  if (digitalRead(wire3) == LOW && digitalRead(wire4) == LOW && operatorEntered) {
    operatorEntered = false;
    digitalWrite(equalSign, LOW);
    digitalWrite(negativeSign, LOW);
    digitalWrite(addition, LOW);
    digitalWrite(subtraction, LOW);
    digitalWrite(multiplication, LOW);
    digitalWrite(division, LOW);
  }

}
```

✅ Thank you for scrolling all the way down here!