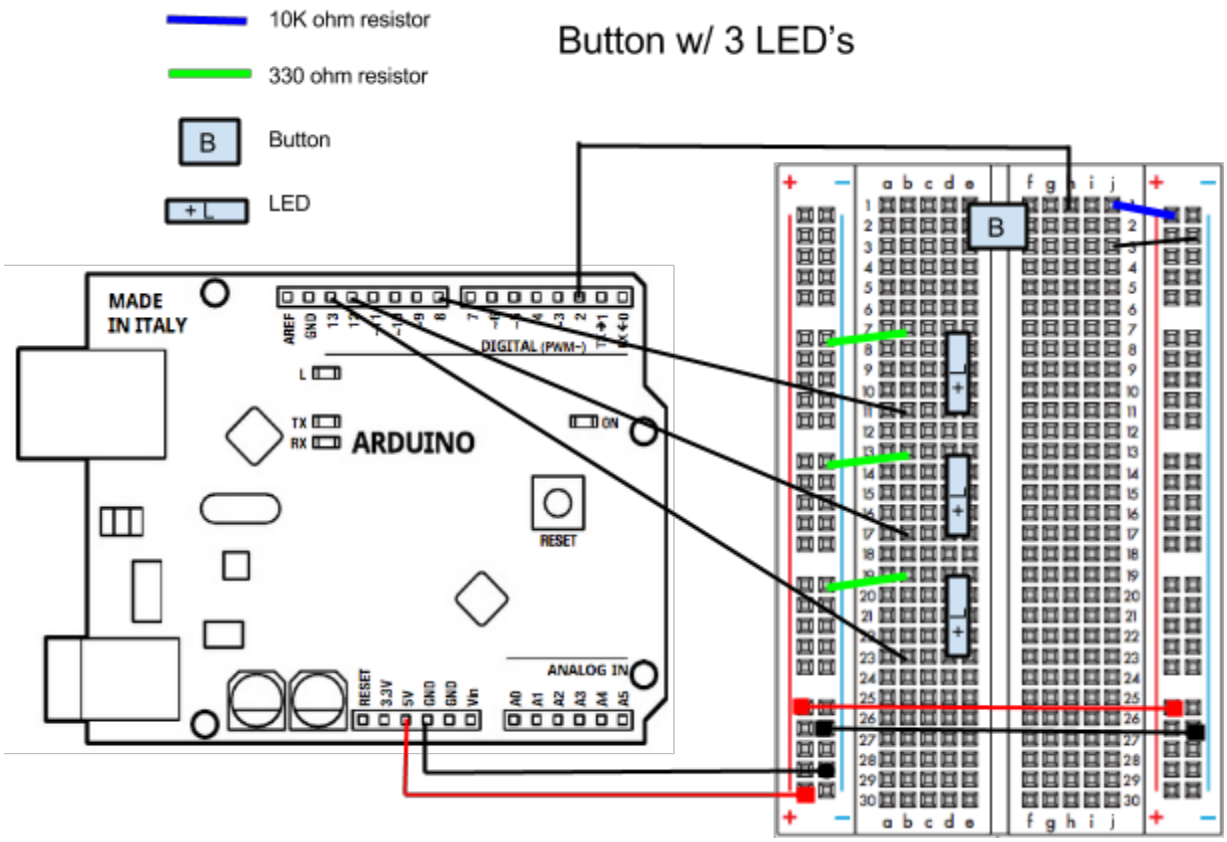
Name: Number Identifier

If you press a button (x) times,
---turn on the primeLED if (x) is a prime number.

---turn on the even LED if (x) is an even number.
---else, turn on the odd LED because its odd if it isn't even.

Materials:
-One 10K ohm resistor connected to the button...Note: It is a Pull-Up Resistor if it's connected directly to the + terminal thus the Current State is LOW.
-Three 330 ohm resistors connected to Three LED's



DECLARE

```
//Assign the pins
const int buttonPin = 2;
const int evenLED = 13;
const int oddLED = 12;
const int primeLED = 8;

//These are used as delays
int wait = 100;
int reset = 1000;

//This is the value of the input...it will increase by 1 everytime the button is pressed
int buttonCount = 0;

//These are both digital values can be (HIGH or LOW) or also written as (0 or 1).
//These allow to detect if the button was pressed by checking the current instance with the previous instance.
int currentState = 0;
int previousState = 0;

//This is an array of numbers, we can modify this array by adding additional numbers.
int primeSet[] = {2,3,5,7};

//This uses a special "sizeof" code...This counts the number of values in the primeSet[].
int primeSize = sizeof (primeSet) / sizeof(int);
```

SETUP

```
void setup()
{
  pinMode(buttonPin, INPUT); //press the button (x) of times
  pinMode(primeLED, OUTPUT); //turn this on when x is prime
  pinMode(evenLED, OUTPUT); //turn this on when x is even
  pinMode(oddLED, OUTPUT); //turn this on when x is odd
  Serial.begin(9600); //we want to print and review real-time data. Very useful for troubleshooting.
}
```

LOOP

```
void loop()
{
  currentState = digitalRead(buttonPin); //is the button pressed? Lets read it using the buttonPin.

  if (currentState != previousState) //if the button state has changed..meaning its NOT like it was before.
  {
    if (currentState == LOW) //Note: I used a pull-up resistor so its set as (LOW) when its pressed.
    {
       if (buttonCount >= 10) //When the buttonCount reaches 10...send the user a RESTART signal.
      {
       buttonCount = 0; //RESTART the buttonCount back to zero.

       //Display a RESTART signal so that the user knows they are starting over.
       digitalWrite(primeLED, HIGH);
       digitalWrite(evenLED, HIGH);
       digitalWrite(oddLED, HIGH);
       delay(reset);
       digitalWrite(primeLED, LOW);
       digitalWrite(evenLED, LOW);
       digitalWrite(oddLED, LOW);
       delay(reset);
       digitalWrite(primeLED, HIGH);
       digitalWrite(evenLED, HIGH);
       digitalWrite(oddLED, HIGH);
       delay(reset);
       digitalWrite(primeLED, LOW);
       digitalWrite(evenLED, LOW);
       digitalWrite(oddLED, LOW);
      }
      else //If the buttonCount is less than 10, then Arduino will perform specific functions.
      {
       buttonCount++; //increment the buttonCount by 1. This is an equivalent statement (buttonCount += 1;)
       Serial.print("Button Count = "); //Print a string
       Serial.println(buttonCount); //Print the current value of the buttonCount

       //turn all of the LED's off. This indicates a buttonCount = 0...where it is neither prime, even, or odd.
       digitalWrite(primeLED, LOW);
       digitalWrite(evenLED, LOW);
       digitalWrite(oddLED, LOW);


       //CONTINUED inside “else”
```

```
        //CONTINUED inside “else”

        //This code allows you perform a specific function if the condition "is_prime(buttonCount)" is True
        if (is_prime(buttonCount))
         {
          digitalWrite(primeLED, HIGH);
         }
        else
         {
          digitalWrite(primeLED, LOW);
         }


        //This code allows you perform a specific function if the condition "is_even(buttonCount)" is True.
        if (is_even(buttonCount))
         {
          digitalWrite(evenLED, HIGH);
          digitalWrite(oddLED, LOW);
         }

        //Simple math, if the number isn't even...then it must be odd.
        else
         {
          digitalWrite(evenLED, LOW);
          digitalWrite(oddLED, HIGH);
         }
       }
     }

  previousState = currentState;
  //This helps you save data of whether the button is being pressed or not.
  }
  delay(wait);
}


boolean is_prime(int inputNumber) {
  for (int i = 0; i < primeSize; i++) {
     if (inputNumber == primeSet[i])
     {
         return true;
     }
  }
  return false;
}


boolean is_even(int n) {
  return (n % 2 == 0);
}
```

/*

---

boolean is_even(int inputNumber)

This uses a modulus technique that calculates the remainder of division.
If the remainder is equal to zero then the number must be even.

We could check to see if n is odd using:

(n % 2 == 1); b/c if n = 11 then (11 / 2 == 5 with remainder 1)

---

boolean is_prime(int inputNumber)

This code is designed to check if some "inputNumber" can be found in an the "primeSet[]" array.
In this case, we are using the buttonCount as the inputNumber.

From above, if (is_prime(buttonCount))

---

for (int i = 0; i < primeSize; i++)

We will use this _for loop_ to test all the values in the array (one by one).
The _for loop_ uses an integer "i" which represents the index of the array.
When the index is zero (i=0)...its referring to the 1st value in the array.

int primeSet[] = {2,3,5,7};
Thus
primeSet[0] = 2, primeSet[1] = 3, primeSet[2] = 5, primeSet[3] = 7, and primeSet[4] = not in the array.

"i < primeSize" means that it will count through all the elements in the array.
"i++" means increment the value of i by 1.

---

if (inputNumber == primeSet[i])

This if statement checks to see if the inputNumber (again the buttonCount) is equal to
any of the values in the array for each index.

Since this statement uses the inputNumber from a boolean. It will return either True or False.
In this case, we want something to happen if the inputNumber is prime so we assign it True.
From above,

```
if (is_prime(buttonCount))
        {
         digitalWrite(primeLED, HIGH);
        }
       else
        {
         digitalWrite(primeLED, LOW);
        }
```

otherwise, it should return False because the inputNumber is nonprime.

This code is an alternative method of checking if the buttonCount is a prime number.
It uses a while loop to perform the exact same task as the boolean above.

```
boolean is_prime2(int inputNumber)
{
            int i = 0;
            while (i < primeSize)
       {
              int currentPrimeNumber = primeSet[i];
              if (inputNumber == currentPrimeNumber)
              {
                        return true;
              }
              i++;
      }
return false;
}
```

---

*/