Speedy blinker

ENGR102 Intro to Electronics

Shoreline Community College

October 2017

Description of the problem. Why are you building this circuit?

The circuit #1 presented on the SIK Guide titled “Your First Circuit: Blinking a LED” is pretty rewarding first simple circuit and very straightforward, it consists of a LED that blinks with 1 second (1000 milliseconds) off/on time as stated in the code. It would be very interesting to find out how long must be the “delay” (on/off time) for a specific person to perceive the LED as blinking purely with their eyeball.

What things can you do with this circuit?

Measures the length of the on/off time of a blinking LED for the user to notice that it is being turned on and off instead of being always on.

Knowledge from which prior experiments is required?

It is only required to know circuit #1 of the SIK guide and how to use a push button which is not covered on the guide until circuit #5, however, the website Arduino.cc (listed in sources) is a very helpful website which has a very easy to follow tutorial on push buttons. In order to deliver the results to the user it is helpful to know how to use the built-in Serial Monitor, the website instructables.com holds a very good guide on Serial Monitor.

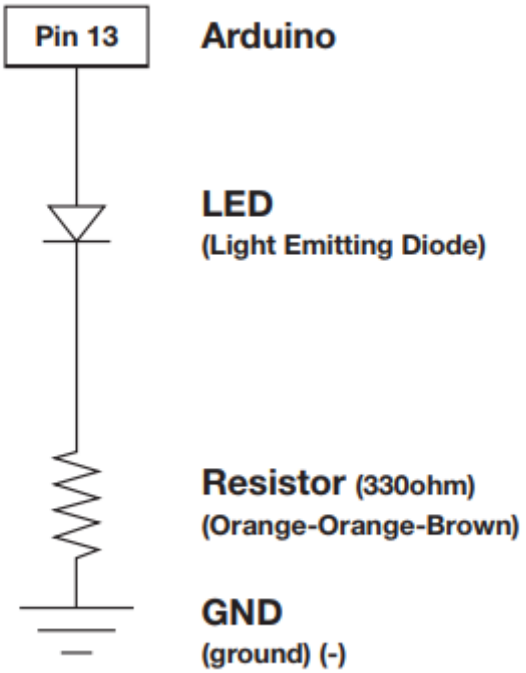

Figure 1. Circuit #1 Schematic. (Arduino SIK Guide)

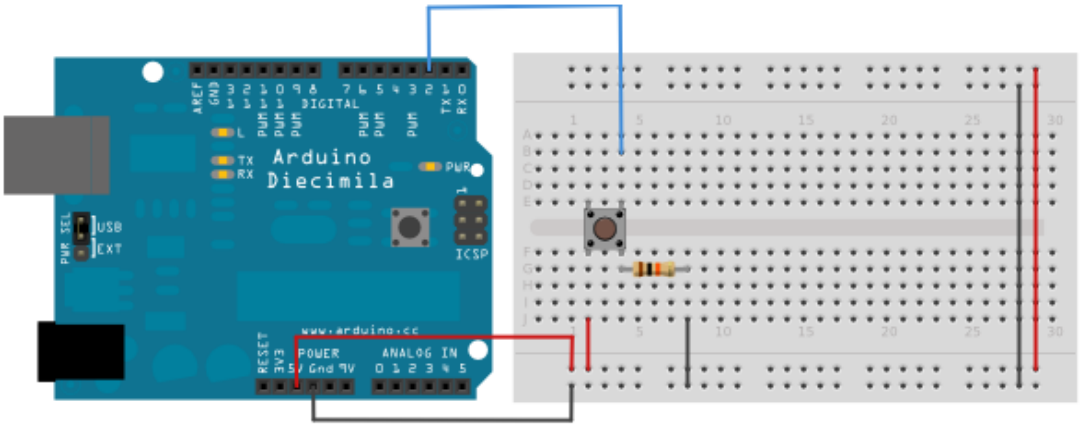

Figure 2. Depiction of a simple push button circuit. (Arduino.cc)

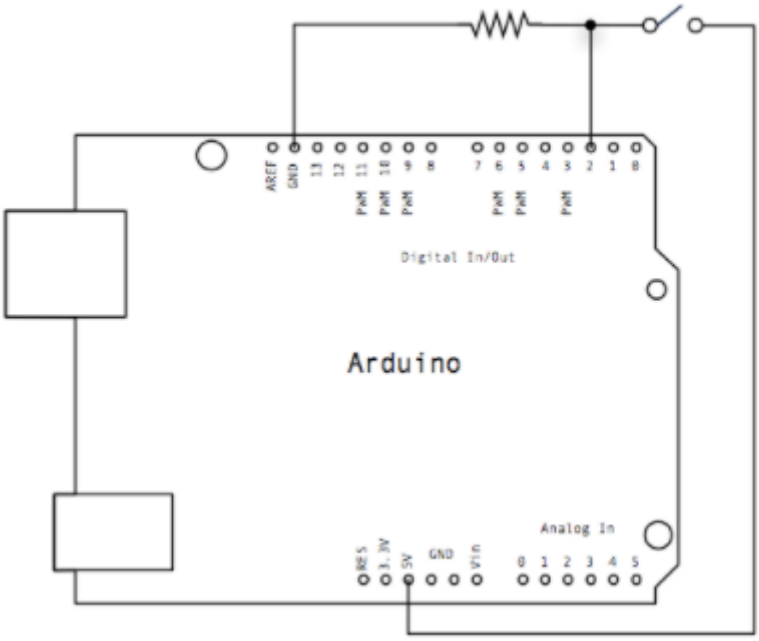

Figure 3. Schematic of the circuit on Figure 2. (Arduino.cc)

The code in the box is for the circuit from Figure 2. And Figure 3. (This is an example on how to use a button, later on the report I have copied the code of my program). (Arduino.cc)

```
/*
  Button

  Turns on and off a light emitting diode(LED) connected to digital pin 13,
  when pressing a pushbutton attached to pin 2.

  The circuit:
  - LED attached from pin 13 to ground
  - pushbutton attached to pin 2 from +5V
  - 10K resistor attached to pin 2 from ground
```

```
  - Note: on most Arduinos there is already an LED on the board
    attached to pin 13.

  created 2005
  by DojoDave <http://www.0j0.org>
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

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

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Parts Required

- One (1) Arduino Board from the Sparkfun kit.
- One (1) Push Button.
- One (1) Color (red) LED.
- One (1) 330Ω Resistor.
- One (1) 10kΩ Resistor.
- Jumpers (Wire).

Problem-solving approach

The approach is quite simple. Knowing how to turn on and off a LED then all that is left is to know is how to control the delay time and the flow of control.

1) Learn how to do the Circuit #1 from the SIK Guide
2) Research on how to use a push button with Arduino
3) Combine Knowledge from 1) and 2) to make this circuit
4) The logic is quite simple too: Power a LED by a digital PIN that we can control with Sketch code, then independently read the state of the push button and interpret as input which will guide the control flow of the program.

Problems

- Picking the initial delay time delayTime: One may very well begin with delayTime = 0, the issue is that one cannot measure less than a millisecond as a limitation by the Arduino board, and even if it were possible the reaction time average of a person is 0.25s = 25ms (by backyardbrains.com) to a visual stimulus which means the Arduino must stay in this state for at least that amount of time for someone to press the button at delayTime = 0.

  Suppose the Arduino can execute a switch or a delay(0) statement in time t ms then (1 switch/t ms) * 25 ms = (25/t) switches. Unfortunately, running time will vary and this will trigger unexpected behavior (25/t as t goes to zero approaches infinity). Also, it is very debatable whether the light is blinking at delayTime = 0 (basically turning the LED on and off as fast as the Arduino can handle).

  The solution is to begin with delayTime = 1, because this means the light is indeed blinking, it is possible to measure with the Arduino and it is possible for us to calculate the minimum number of switches for humans to perceive the blinking.
- Bias of the user by seeing the delayTime variable in Serial Monitor: In one of the earliest implementations the program would tell the user what the delayTime is in real time. That turned out to be very useful for debugging purposes but could possibly run into some bias issues. The solution was very simple, to delete that line of code and only inform the user of the results.
- Picking the right number of switches every iteration of the loop: This number (150) was picked because even under the shortest delay. 1ms each delay(), 2ms each iteration of the innermost loop, if the average reaction time of a person is 250ms, take the ratio 250ms/2ms/iteration = 125 iterations, then added 25 just to make sure they can press the button.
- Ending the program: During initial experimentation it had been discovered that the Arduino can’t end execution of a program until it is out of power. The initial solution was very simple, to code an endProgram() procedure that would keep looping doing nothing in special. However, now the solution is much better as there is a testEnded() procedure (instead of endProgram()); it keeps doing nothing visually but is still listening to the push button so that the user can take another test when he is willing to, a very appreciated feature.

- Keeping the stack low: Because loop() would call itself after testEnded() had returned (picture a loop(); statement in the code right after endProgram();, the code is in the next pages) the program would work correctly but deep down it is obvious the stack of function calls would keep rising until the Arduino ran out of memory (after a few thousands of tests). We know that the Arduino will keep running the loop() procedure if it has nothing else to run, even after a return; statement in loop(), therefore changing “loop();” for “return;” managed to keep the stack as low as possible solving the problem.

Schematic

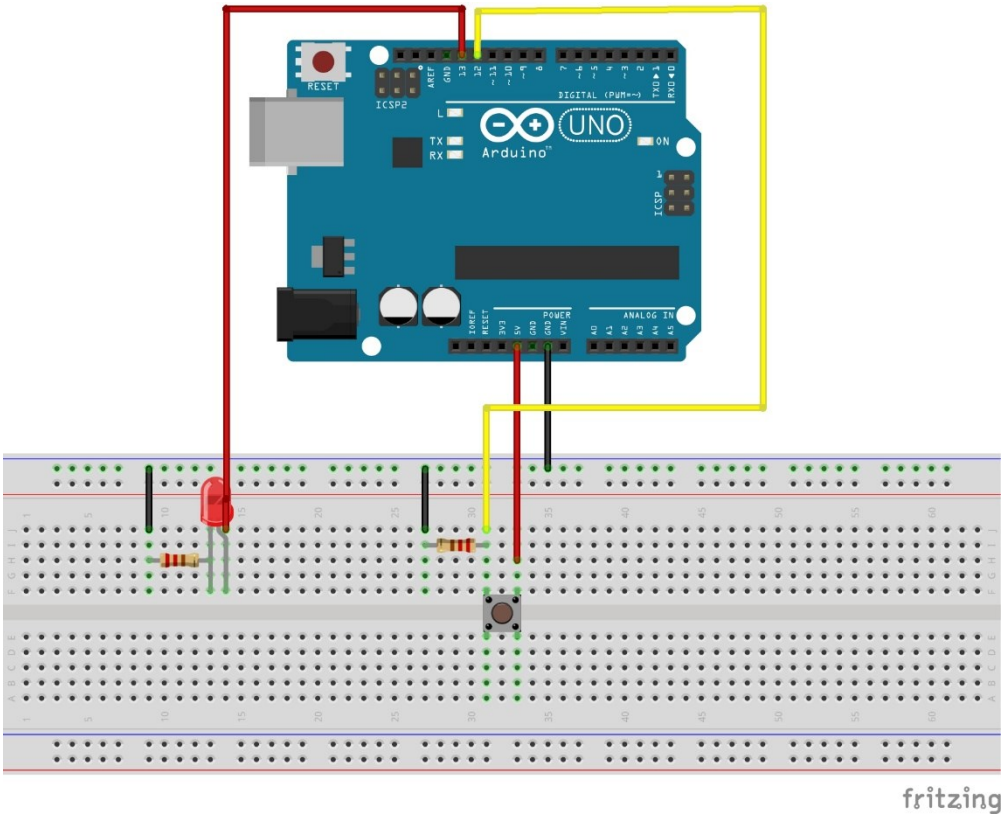

Figure 4. Breadboard view

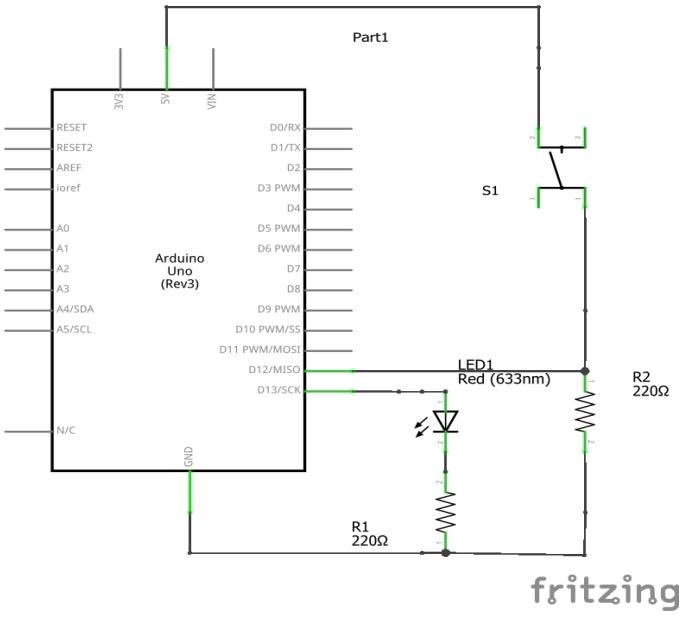

Figure 5. Schematic

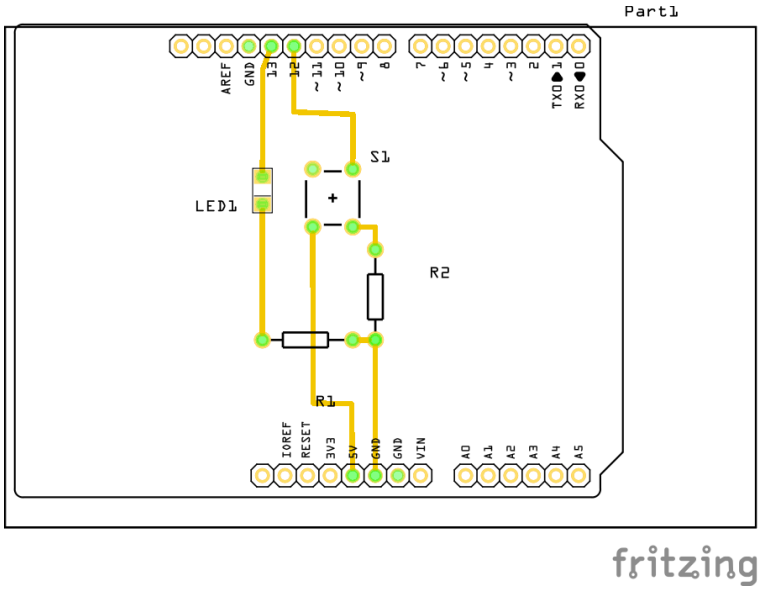

Figure 6. PCB View

Code

```
/*
 * Rafael Laya
 * Professor Basham's ENGR102 class at Shoreline Community College
 * The reader can use this code as they please but I won't be liable of anything
 *
 * Purpose and usage:
 * Blinks a LED connected to pin 13
 * while varying the duration of on/off time
 * when user presses the button at pin 12, serial monitor will output the result
 * for re-testing just press the button again (or you can also press the Arduino reset button)
 */
// Set up some constants for abstraction and portability of the code
const int redLed = 13;
const int pushButton = 12;
int buttonState;

void setup()
{
  pinMode(redLed, OUTPUT);
  pinMode(pushButton, INPUT);
  buttonState = 0;

  // getting serial monitor ready for user interaction purposes
  Serial.begin(9600);
  Serial.println("********** Serial monitor has just been initiated **********");
}
```

```
void loop()
{
  delay(1000);

  // advise the user the test has begun
  Serial.println("New Test initiated");

  // try from 1ms up to 100ms delay in steps of one
  for(int delayTime = 1; delayTime <= 100; delayTime++)
  {
    // blink 150 times during each try
    for(int k = 0; k <= 149; k++)
    {
      buttonState = digitalRead(pushButton);
      blinkRedLed(delayTime);

      // when the user decides he can see the LED blinking he'll push the button
      if (buttonState == HIGH)
      {
        // output the results
        Serial.print("You can see the blinking light when it blinks every ");
        Serial.print(delayTime);
        Serial.println(" milliseconds");
        testEnded();
        return;
      }
    }
  }
  // wait 5 seconds before re-testing (in case the user never pressed the button)
  delay(5000);
}

void blinkRedLed(int delayTime)
{
  digitalWrite(redLed, HIGH);
  delay(delayTime);
  digitalWrite(redLed, LOW);
  delay(delayTime);
}

void testEnded(void)
{
  delay(1000);
  /*
   *  user will press the button after the test has ended
   *  if it is the case that he wants to take another test
   */
  while (true)
  {
    buttonState = digitalRead(pushButton);
    if (buttonState == HIGH)
    {
```

```
      buttonState = 0;
      break;
    }
    else
    {
      continue;
    }
  }
  return;
}
```

The code acts as follows:

1) Starts by taking a one second delay.
2) Prints into the serial monitor “New Test initiated”.
3) Tests the user for delays from 1ms to 100ms in steps of one.
4) Each try it is going to blink 150 times (blink in this particular case means to turn the LED on, delay, off, and delay again).
5) As soon as the user presses the push button they will receive their results.
6) Program will keep listening to the push button waiting for the user to press and then begin a new test.

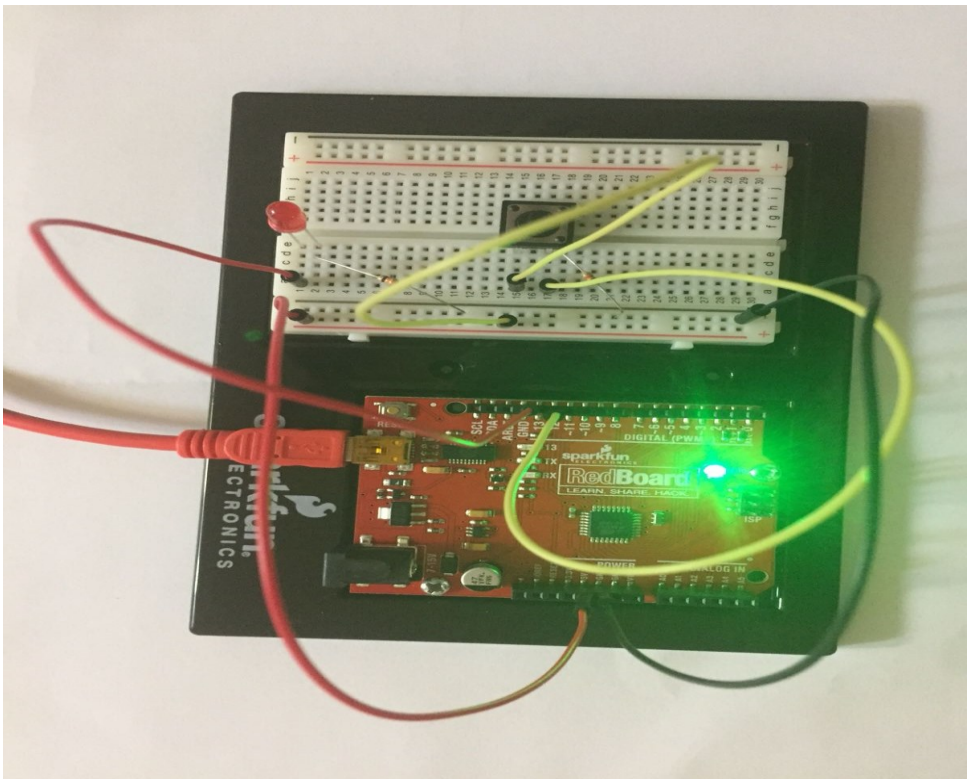

Figure 7. Photo of the circuit

Data

The following charts summarize the experience of testing users with the yellow, blue and red LEDs (The primary colors). Each user had two chances (or more if requested) per test, only the last trial was recorded. Subjects were also

tested under conditions of light (white light) and obscurity (No lights). In total, each user had six tests to perform.

- Fixing the Light Conditions and varying the Colors:

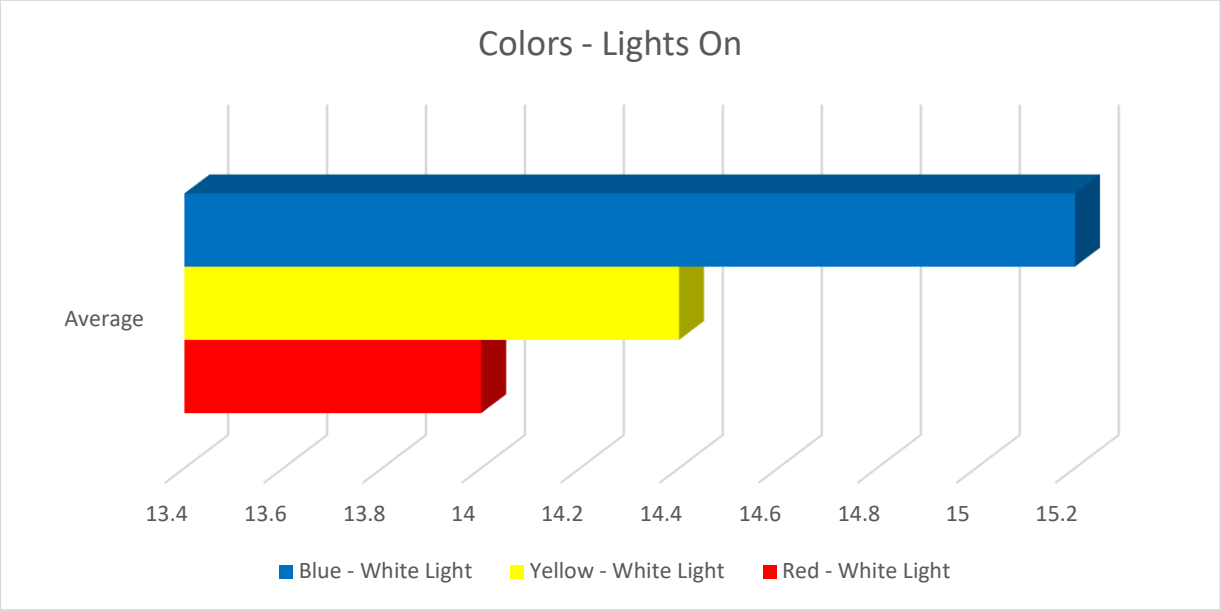

Figure 8. Colors – Lights On

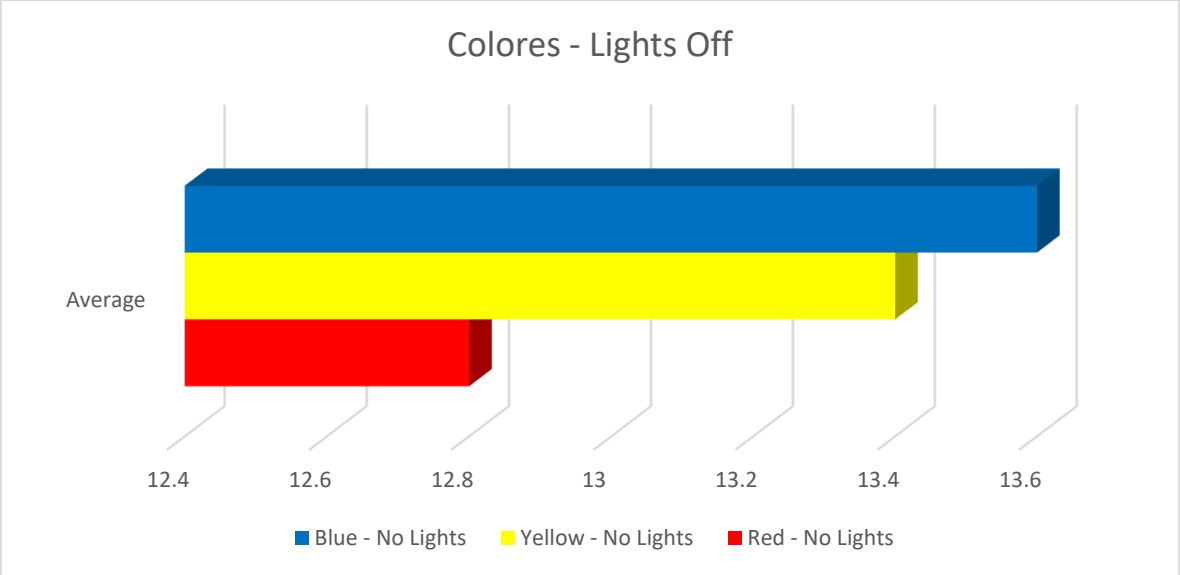

Figure 9. Colors – Lights Off.

These two graphs suggest:

1) In both light conditions the user is more likely to press the button first if the color of the LED is red, second if the color is yellow and third if the color is blue.

One possible explanation can be backed up by this paper from the Rochester University from June of 2011 briefly explained on sciencedaily.com: https://www.sciencedaily.com/releases/2011/06/110602122349.htm

From its summary: “When humans see red, their reactions become both faster and more forceful. And people are unaware of the color’s intensifying effect, according to a new study.”

From visual.ly the user Paul Van Slembrouck (https://visual.ly/blog/the-use-of-yellow-in-data-design/) explains the sensitivity of the eye regarding colors in terms of the architecture of the eyes. Turns out human eyes are more sensitive to yellow in terms of intensity, and therefore it may have played a small role in the results (yellow vs blue). However, the effect of enhanced reaction explained on the University of Rochester’s paper played a higher role in the experiment as the results were supposed to test reaction more than perception of the intensity (although intensity is definitely a factor of the reaction time, explained by the yellow vs blue results).

- Fixing the Colors, changing the light conditions:

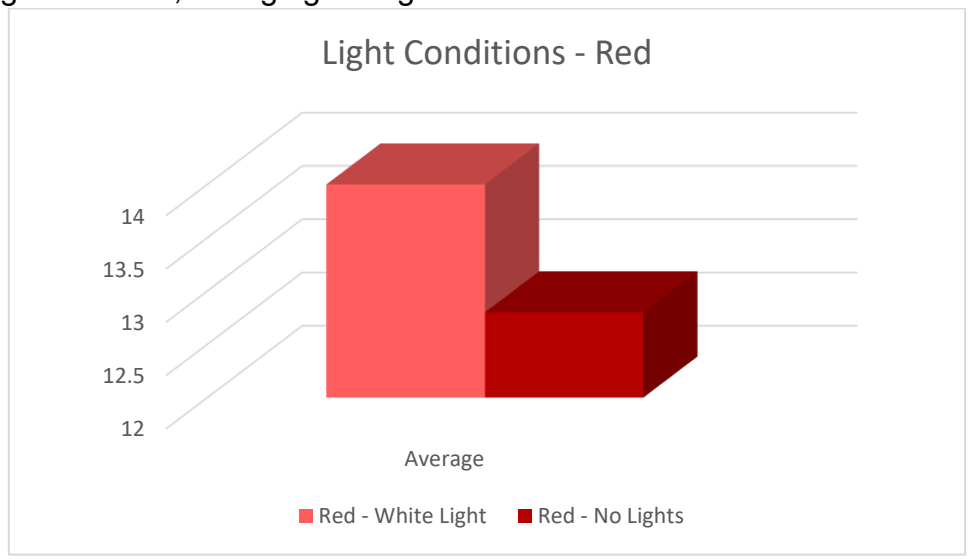

Figure 10. Red – White Light vs Lights Off

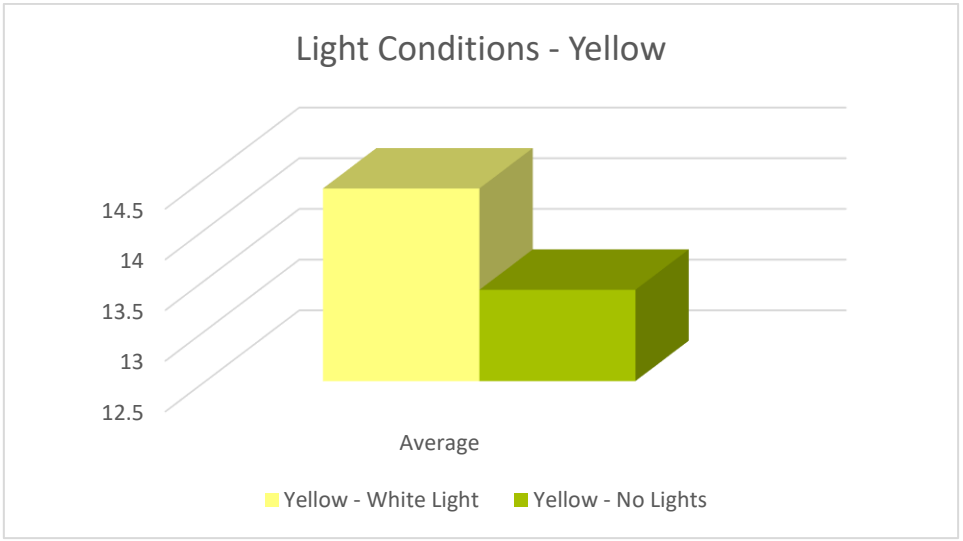

Figure 11. Yellow – White Light vs Lights Off

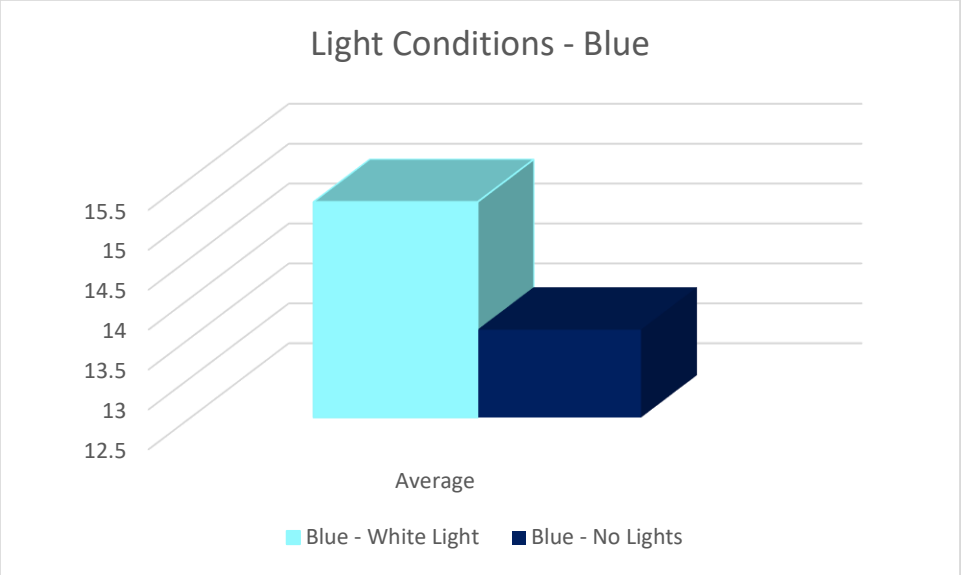

Figure 12. Blue – White Light vs Lights Off

We can observe:

1) The trend of the colors has not changed, which is expected.
2) Users tend to press the button faster in the darkness.

The result (2) suggests that it is easier for a user to detect the light as blinking when there are no surrounding lights to confuse him. Pick for instance a blue LED under white light and it will be seen as blue, turn the lights off and you are likely to not see the LED at all.

The following tables contain the raw data obtained during the testing:

Subject Number	LED Color	Light Condition	Result (in ms)
1	Red	White Light	13
2	Red	White Light	15
3	Red	White Light	15
4	Red	White Light	13
5	Red	White Light	14
Average			14

Subject Number	LED Color	Light Condition	Result (in ms)
1	Yellow	White Light	14
2	Yellow	White Light	13
3	Yellow	White Light	16
4	Yellow	White Light	14
5	Yellow	White Light	15
Average			14.4

Subject Number	LED Color	Light Condition	Result (in ms)

1	Blue	White Light	16
2	Blue	White Light	14
3	Blue	White Light	15
4	Blue	White Light	14
5	Blue	White Light	17
Average			15.2

Subject Number	LED Color	Light Condition	Result (in ms)
1	Red	No Lights	12
2	Red	No Lights	13
3	Red	No Lights	13
4	Red	No Lights	12
5	Red	No Lights	14
Average			12.8

Subject Number	LED Color	Light Condition	Result (in ms)
1	Yellow	No Lights	13
2	Yellow	No Lights	12
3	Yellow	No Lights	14
4	Yellow	No Lights	13
5	Yellow	No Lights	15
Average			13.4

Subject Number	LED Color	Light Condition	Result (in ms)
1	Blue	No Lights	12
2	Blue	No Lights	15
3	Blue	No Lights	14
4	Blue	No Lights	13
5	Blue	No Lights	14
Average			13.6

Resources

https://backyardbrains.com/experiments/reactiontime

https://www.arduino.cc/en/Tutorial/Button

https://cdn.sparkfun.com/datasheets/Kits/SFE03-0012-SIK.Guide-300dpi-01.pdf

https://www.instructables.com/id/HOW-TO-use-the-ARDUINO-SERIAL-MONITOR/

https://www.sciencedaily.com/releases/2011/06/110602122349.htm

https://visual.ly/blog/the-use-of-yellow-in-data-design/