***Etude 02 – LoopyLoop***

**Part One:**

I started with grouping out the components needed for the keyboard circuit. I am starting to differentiate better the resistors between each other. I started a personalized legend to not forget the differences between resistors. I grouped the needed resistors to their own set and wrote a little note to differentiate each group. Once I brought all the needed components together, I started to build up the circuit by following the steps brought in the etude 02’s PDF. For this etude, I was more comfortable to build the circuit; I was able to build it in a shorter amount of time.

Une image contenant texte

Description générée automatiquement

A picture of the first step: grouping components.

Une image contenant texte, équipement électronique

Description générée automatiquement

A picture of the second step: connecting RGB LED to the breadboard.

Une image contenant texte, équipement électronique

Description générée automatiquement

A picture of the third step: adding buttons to the breadboard.

Une image contenant texte, équipement électronique

Description générée automatiquement

A picture of the fourth step: adding Piezo buzzer to the breadboard.

Une image contenant texte

Description générée automatiquement

A picture of the fifth step: connecting mode button to the breadboard.

Une image contenant texte

Description générée automatiquement

A picture of the sixth step: connecting missing wires to the breadboard and voltage pins on the Arduino.

The most complex part was to build the code. I had a hard time to write the on my own. It was easier for me to write the code by following the teacher during classes than building it up on my own. I had to watch recordings of past classes and to look at Arduino API in order to find ways to write the code.

I wanted to test if the program was functioning correctly.  It was hard to find a reflective surface to see perfectly the words appearing with the Perceptron. I tried with a TV screen surface, and it worked. I was able to see the words while moving the breadboard.  I needed a lot of darkness in the room to correctly see the words appearing on the surface. I needed to make broad movements at a certain speed to see the words correctly.  I had a hard time moving my breadboard. The components fell a few times due to my rough movements.

A picture of myself testing the Perceptron-P.

Second picture of myself testing the Perceptron-P.

Third picture of myself testing the Perceptron-P.

**PART TWO:**

The visible features that distinguish both circuits are their type of circuits and their number of black wires and resistors.

While testing the circuits, I noticed how the lights were blinking irregularly and asynchronously in the built circuit. The amount of voltage was unequal between the LEDs.

In the alternate circuit, the voltage in LEDs’ lights was well-balanced. The voltage distribution was equal among LEDs, which resulted in a similar pace, speed, and lighting in their blinking.

After observing the scheme of the circuits and these tests, I concluded that the built circuit is in parallel while the alternate circuit is in series. Each LED has its connection to a resistor and wires in the built circuit. The distribution of voltage from the Arduino microcontroller goes into the separate wires, and the resistors decrease their amount of voltage and current. This decrease results in a weaker light and changing speed of blink through the LEDs.

In contrast, the alternate circuit has only one resistor. It results in a high current and amount of voltage into connected black wires. The wires transmit the same current into the circuit; there is no division in the voltage. These characteristics explain the synchronized blinking and intense lighting into the LEDs when the circuit is working.

In this context, the built circuit is the most reliable circuit to create the Perceptron-P. If a wire detaches itself from the breadboard, the whole circuit will still work due to the separated paths of wires, distributing the current equally in this circuit. On the contrary, if a wire detaches itself from the breadboard in the alternate circuit, the Perceptron-P will completely stop working due to being a circuit in series. The current is not distributed equally into separate paths. The current flows through a linear path created through wires' connections. The current will never reach the end of the circuit if a component is not working correctly, causing an immediate stop through the whole circuit.

**PART THREE:**

1. I tried to figure out the button's use while testing the Perceptron-P. I noticed it serves at resetting the program written in the Arduino text editor. The word reappears from the start after pressing it.
2. The reset function serves to see the message from the beginning if needed. A long black wire links the button's row and the reset hole pin in the Arduino microcontroller on the circuits. The resistor inserted below the long wire decreases the voltage and current when the button activates. The reset pin in the Arduino microcontroller receives the rest of the current. A shorter black wire connected below the edges of the button serves to deplete life signals transmitted into the button. This action nullifies the voltage and current into the circuit.

**PART FOUR:**

For the custom message, I wanted to keep it simple. I wrote “Arduino!” because this circuit uses the Arduino microcontroller and its text editor. I tried my custom message on both circuits, and it worked. It was really fascinating to see how open the options are to make words or sentences appear with the Perceptron-P.

A picture of myself testing the Perceptron-P with my customed message.

Second picture of myself testing the Perceptron-P with my customed message.