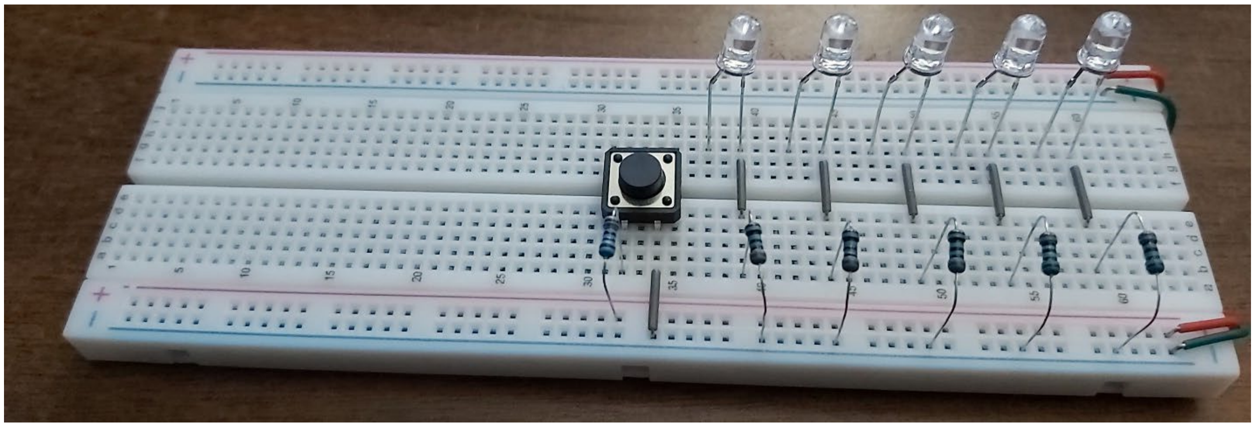
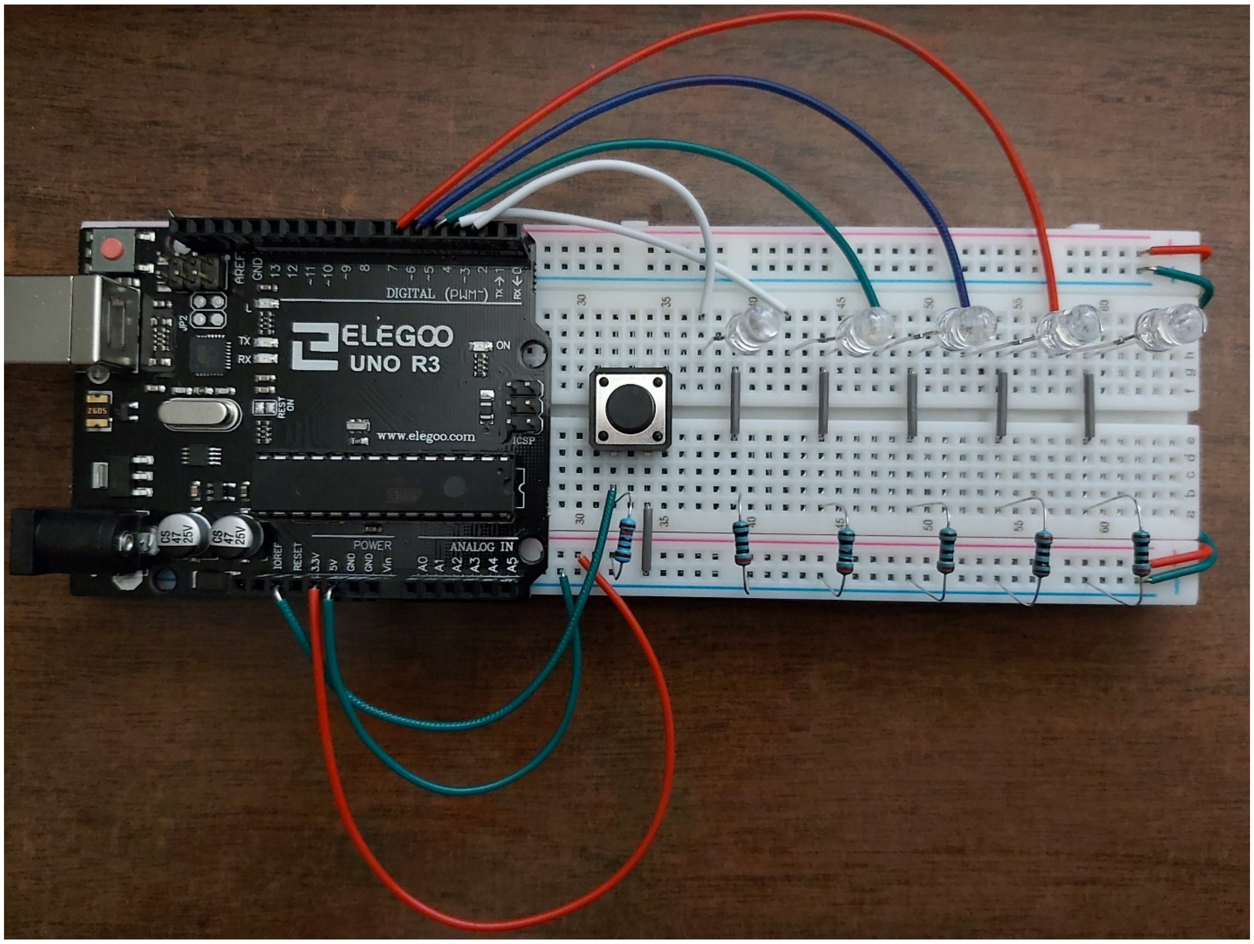
Etude 2: Perceptron-P

Melissa Lim
23/10/2020

PART ONE: Percepton-P (Etude-Two Circuit to be Built)

I started by adding the components on the breadboard. The Arduino microcontroller was added in the next step with the wires connected to the breadboard. I then connected the Arduino to my computer and uploaded the program that was given for this etude. When the program ran through my Arduino, all except one LED didn't light up. I then noticed it was a defect in my breadboard. There is one pin that is unable to conduct the current for some reason… Therefore, I had to move everything to the right and things worked out correctly this time around. I unplugged the USB and plugged in the battery and that is how my portable circuit was built.



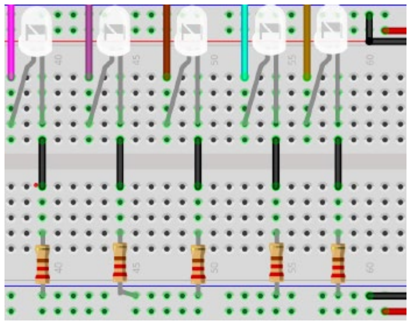


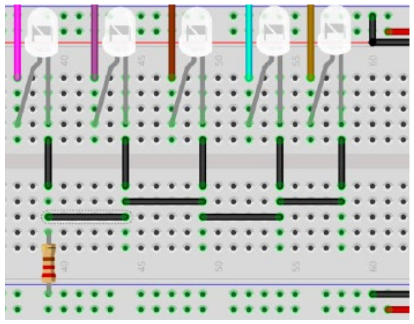
One of the things I noticed after building the circuit was that if you removed the red wire connected to the 5V output of the Arduino, the button on the board is still functional. My hypothesis is that the green wire connected to the "RESET" port accesses the system's voltage output as well.

PART TWO: Perceptron-P (Etude-Two Alternative Circuit)

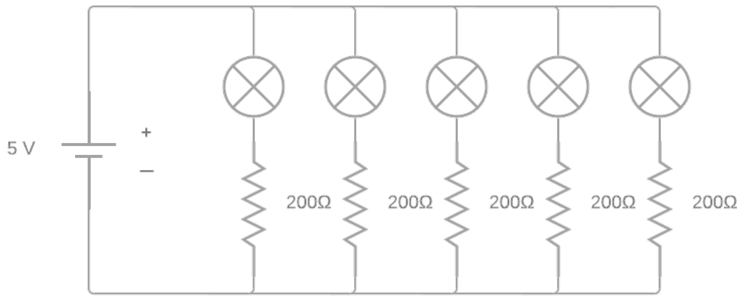
*Compare and contrast the Fritzing circuit illustrations for the **Built Circuit** to the **Alternate Circuit**.* In the areas that differ between the two circuits, there is only one resistor used in the alternate circuit while there is 5 in the built circuit (one resistor per LED light).

Built Circuit   vs   Alternate Circuit

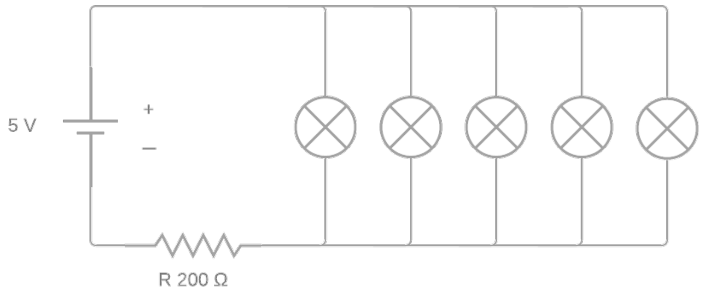




I made simplified diagrams to showcase the difference between the two, only focusing on the areas of concern. The battery symbol represents the Arduino I/O of current and ⊗ represents LED lights.



Built Circuit



Alternate Circuit

The built circuit represents a parallel circuit while the alternate circuit represents a circuit in series. Since every component (resistors + LED) share two nodes in the built circuit, it is a circuit in parallel. The resistors change the way the current flows through both circuits.

The most reliable circuit would be the one in parallel because the voltage across each LEDs will be fixed and they will all shine as bright, while a circuit in series would make the voltage vary depending on how much each energy each LED uses.

To calculate the current flow in the parallel (built) circuit:

$$I = \frac{V}{R} = \frac{5\ \mathrm{V}}{200\ \Omega} = 0.025\ \mathrm{A} = 25\ \mathrm{mA}$$

Since there are 5 resistors in the circuit:

$$I = I_1 + I_2 + I_3 + I_4 + I_5 = 25\ \mathrm{mA} + 25\ \mathrm{mA} + 25\ \mathrm{mA} + 25\ \mathrm{mA} + 25\ \mathrm{mA} = \mathbf{125\ mA}$$

To calculate the current flow in series (alternative) circuit:

$$I = \frac{V}{R} = \frac{5\ \mathrm{V}}{200\ \Omega} = 0.025\ \mathrm{A} = \mathbf{25\ mA}$$

In conclusion, there is less current flowing through the alternative circuit, therefore it is less efficient.

PART THREE: Perceptron-P

a) What does [the highlighted area] do?

When the button is pressed, the whole system resets, therefore the lights stop and the code is re-run; after a few seconds the lights turn back on again. It works the same way as the red button on the Arduino microcontroller does.

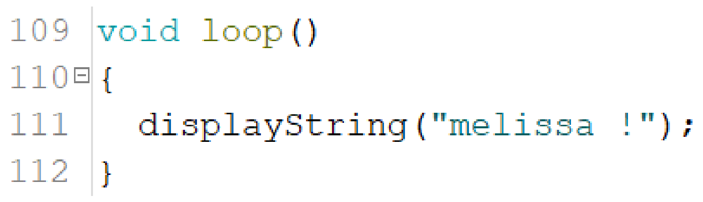
b) Why/how is the functionality implemented (electrically)?

From the Arduino, we get an input of 5 Volts (5V) from the red wire and an output to ground (GND) from the black wire at the bottom left. The 5V current is connected through the breadboard to a 10k Ω resistor, passing through another black wire that connects to the "RESET" port of the Arduino and finally connects to a button. The other end of the button is connected by a small black wire to the ground line in the power rail of the breadboard. This line in the breadboard connects to a wire accessing the GND (ground) and represents the output of electricity. Therefore, when this button is pressed, it activates the RESET function of the Arduino which re-runs the code. It is like plugging and unplugging the battery.

PART FOUR: Create a Unique Perceptron-P Message

Every character is implemented at the beginning of the code with arrays of integers. The displayLine(int line) takes information from the array index to light up specific LEDs at a specific time. Since the creation of the characters is already handled by the displayChar() function, to add a new message we only need to replace the string being read in the loop() function at the end.

The displayString() function is going to read the string argument, character by character, and display each character by referring to the displayChar() function.



```
void loop()
{
  displayString("melissa !");
}
```

To be honest, it was really difficult for me to see and capture the message on the board. I wonder if it’s because my LEDs weren’t close enough or I didn’t wave it correctly, but I couldn’t figure out the message clearly. I tried recording a video with an app that allowed me to lower my shutter speed but the outcome is still unreadable.