Cart 360
Samuelle Bourgault
November 2nd, 2018

ETUDE 3

PART ONE

The build of the circuit was pretty straightforward. I realized that I didn’t know which branch of the LED was the red, green and blue, but that I would figure out when implementing the code.

On selectMode() function:
The first version of that function was skipping some modes the longer I was pressing on the button. So I decided to add a new variable to register the previous buttonState. It works fine but I realized that the instruction at the beginning of the code says that it is not needed to add variables or constants. So I added a short delay after each change of mode, 300 ms seemed working well with my subjective way of pressing buttons.

On setRGB() function:
So I realized that my first assumption on the color of each pin of the LED was off. Initially, I had this:

```
#define LED_PIN_R 5 // R PIN
#define LED_PIN_G 9 // G PIN
#define LED_PIN_B 6 // B PIN
```

But switched it for:

```
#define LED_PIN_R 9 // R PIN
#define LED_PIN_G 6 // G PIN
#define LED_PIN_B 5 // B PIN
```

On reset() function:
I am just resetting each element of the array to zero.

On live() function:
The last note sounds like shit. Hopefully, part two of this etude will lead to a better sound.

On record() function:
I realized that it’s important to add some delay between each stored note. Otherwise, it doesn’t take into account the time required for the user to press and release the button and record, therefore, a series of the same note in the array.

On play() and looper() functions:

I decided to add a noteAt variable in order to iterate through the array of notes. Only one note is being played per loop. I found that this method makes the change of modes easier than using a while or a for loop and to break from it when the mode button is being pressed.

On looper() function:
I decided to make my looper() function go back and forth through the array.

Without the duration parameter in the tone function. The tone just never ends until you get to the live mode and press a new note.

On analogRead() function:
The outputted values are 7, 82, 486, 931, 1023, but vary slightly depending on how the button is pressed. This information will be useful in part 2 of the etude.

PART THREE

The ladder involves five different resistances (10kΩ, 1kΩ, 10kΩ, 100kΩ and 1MΩ) connected in series. Each time the user presses on a button, it allows the current coming from the 5V Arduino output to flow toward the ground passing through between one to five resistance. The analog reading is taken before the first 10kΩ resistance that is directly connected to the ground. Therefore, when the first button is pressed, the analog reading returns 1023, the numerical equivalent to 5V since there is no drop of potential between the 5V source and the point of reading. When the second button is pressed, the current passes through the 1kΩ resistance before being read. Voltage division occurs between the 10kΩ resistance connected to the ground and the first 1kΩ resistance, but the drop in potential is small since 1kΩ is ten times smaller than 10kΩ. The returned analog value is around 931. When the third button is pressed, the current passes through 11kΩ before being read which returns a value of 486. The same goes for the fourth and the fifth buttons that allow the current to pass through 111kΩ and 1111kΩ (1.111MΩ) and output values of 82 and 7 respectively. These specific values can then be associated with different events in the Arduino code.

The mode selector involves a button connected to the Arduino 5V output and the ground through a 10kΩ resistance. A digital reading is being done at the exit point of the button. When the button is open, the digital reading is 0V or LOW as expressed in the code. When the button is pressed, the current flows and the exit point of the button reaches 5V. In this state, the digital reading is HIGH. In the program, each time the program sees HIGH, it increases the mode value by one unit, evolving from 0 to 4 and back to 0 once it reaches 4.

Finally, once the input is read. The tone function through a PWM digital pin send equivalent voltage to the piezo electric crystal. This voltage indices mechanical vibrations in the crystal which creates audible waves in the perceptive range for human ears.