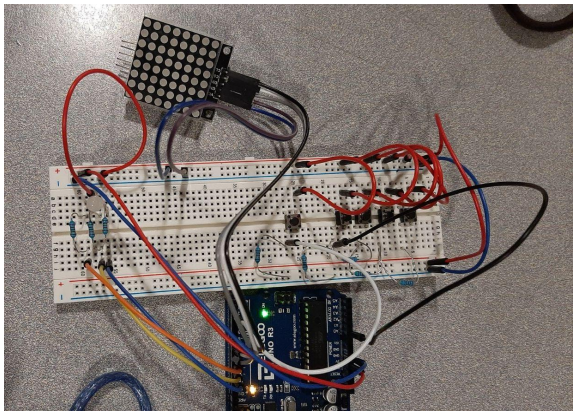
CART 360
Reihaneh Tamizkar
ÉTUDE 3

Documentation

Circuit building: Man Zou and Reihaneh Tamizkar
Overall we did not have to make any significant changes to the circuit except for the RGB LED. The circuit has 4 important parts: the state LED which helps us visualize the state of the program we are in, the matrix which displays the patterns, the state button which switches to the next state at each press, and the resistance chain of buttons which determines which pattern to display on the matrix.



Observation:

- RGB light: our RGB LED has a common cathode, so we had to switch the wire connection from the + side to the - side.
- The resistance buttons placed in a row gave a value of (210-230), (330-350), (500-520) and (900-1020) on the microcontroller's analog input.

Coding the microcontroller: Man Zou and Reihaneh Tamizkar
Part 1: implementing state change.

1. `setMode()`
   To code the setMode, we had to introduce 2 variables called `currentBtnState` and `pastBtnState`. `currentBtnState` reads the digital input from the mode button pin (pin 2) and we have to incorporate a `pastBtnState` to make sure that each time the button signal is high, the `mode ++` command only gets executed once. We also have to add a delay for the code to be executed properly.
2. `setRGB()`
   Coding `setRGB()` is much more straight forwards, all we needed to do is to `analogWrite()` the colour to be turned on at each mode.
3. `runMode()`
   Similar to `setRGB()`, `runMode()` was also very straight forward. We only needed to call the respective function for each mode number.

   Video documentation for part 1: see repo folder.

Part 2: play, record, and loop matrix patterns

1. matrix patterns
   On the .h file that contains the two-dimensional array, we understood that the function `setRow()` from the library loops through all the elements (rows) of the 1st

array element (pattern) one by one, and display the respective light when it reads 1. Understanding that, we designed 4 other patterns on the header file.

2. `play()`
   For the function `play()`, we first had to determine which button on the resistance ladder is being pressed by `analogRead()` the A0 pin. According to the number shown, we can compare it to the number we observed previously and know which button is pressed. We are using `countpattern` as the value that represents the index of the splat array, which is the pattern to be shown. We are also using `resistanceBtn` to store the information of the button that has being pressed to use it for the `switch()` statement. In the switch statement, we simply call the function `drawPatternByRow()` to display the pattern associated to the pressed button.
3. `Record()`
   For the function `record()`, we are doing the buttonpress check similar to `play()`, except now we are also increasing the index for the pattern array if the button is being pressed (`resistanceBtnOn`). We shut down `resistanceBtnOn` to false immediately after the `patternsIndex ++` and add a `delay()` to prevent any undesired looping. We then assign the recorded pattern to the respective element in the array. Finally, we display the pressed pattern the same way as we did in `play()`.
4. `Loopop()`
   For the looping function, by using a for each loop, we go through every element of the patterns array and draw that element through `drawPatternByRow()`. In the for loop, we had to check if the mode button has been pressed, and if so, change to the mode number and break out of the loop.

   Video documentation for part 2: see repo folder.

Part 3: Add an inverted pattern

1. `drawPatternByRow()`
   For this, we added another array of spats that contains the inversed pattern. And we initialized the pattern version to be the not inversed one. If the initial pattern is played, we add a value to the past pattern, thus making it fulfill the condition for the inversed pattern to play next time. The code is the following:

```
void drawPatternByRowNew(int patternSelect) {
  int currentPattern = patternSelect;
  // CLEAR LED MATRIX
  lc.clearDisplay(0);
  if (pastPattern > 3) {
    for (int i = 0; i < 8; i++) {
      lc.setRow(0, i, ( splats[patternSelect][i]) ); //the initial led pattern
      pastPattern = currentPattern;
      delay(REFRESH_RATE_2); // CHANGE TO INCREASE REFRESH RATE.
    }
  } else if (pastPattern <= 3){
    for (int i = 0; i < 8; i++) {
      lc.setRow(0, i, ( splatsInverse[patternSelect][i]) ); //the reverse led pattern
      pastPattern = currentPattern +4;
      delay(REFRESH_RATE_2); // CHANGE TO INCREASE REFRESH RATE.
    }
  }
  delay(HOLD_PATTERN); // CHANGE TO INCREASE REFRESH RATE.
  // CLEAR LED MATRIX
  lc.clearDisplay(0);
}
```

Video documentation for part 3: see repo folder.