Embedded pool

Day 03 : RGB & advanced timers

contact@42chips.fr

Summary: Electro and unicorn vomit

Chapter I

Introduction

Recipe to prepare delicious triangular tortilla-style chips!

In a bowl, mix corn flour, salt, paprika, curry and pepper. Choose the amount of spices according to your taste.

Pour the oil in the center and mix a little. Then pour in warm water and mix the dough with your hands.

Form a ball and let the dough rest for about 30 minutes at room temperature. At the end of the resting time, preheat your oven to 170°C.

Take a silicone mat or a sheet of baking paper the size of your baking sheet. Dust the top with flour. Cut the dough into 2 parts.

Roll out the first part thinly on the mat or sheet, sprinkle with paprika, then roll the dough again.

Using a pizza cutter, form a rectangle and then make vertical and horizontal lines to form squares.

Then pass the cutter diagonally to obtain triangles.

Do not separate the dough and bake as is, the tortillas will separate easily. Do the same with the remaining dough.

Bake the first tray for 15 to 20 minutes. Pay attention to the color of the dough, it can go quickly at the end of the cooking time.

Let cool to room temperature, bake the second batch of tortillas and detach the triangles from the first batch.

The tortillas should be thin and crispy!

Chapter II

General instructions

Unless explicitly stated otherwise, the following instructions will be valid for all assignments.

- The language used for this project is C.
- It is not necessary to code according to the 42 norm.
- The exercises are ordered very precisely from the simplest to the most complex. Under no circumstances will we consider or evaluate a complex exercise if a simpler one is not perfectly successful.
- You must not leave any files other than those explicitly specified by the exercise instructions in your directory during peer evaluation.
- All technical answers to your questions can be found in the `datasheets` or on the Internet. It is up to you to use and abuse these resources to understand how to complete your exercise.
- You must use the datasheet of the microcontroller provided to you and comment on the important parts of your program by indicating where you found the clues in the document, and if necessary, explaining your approach. Don't write long blocks of text, keep it clear.
- Do you have a question? Ask your neighbor to the right or left. You can ask in the dedicated channel on the Piscine's Discord, or as a last resort, ask a staff member.

Chapter III

Campbell's Soup I

	Exercise 01
Red Green Blue	
Turn-in directory : *ex*01/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	
Notes : `n/a`	

You must write a program that controls the RGB LED D5.

- The LED should turn on in red, then green, then blue in a loop.
- Changing every 1 second.

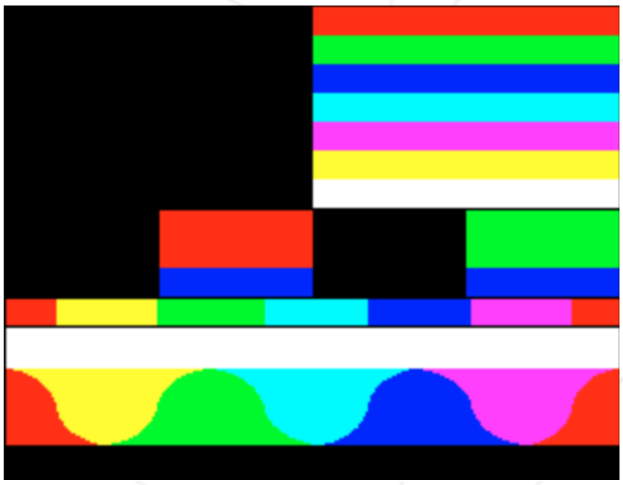
Chapter IV

Technicolor

	Exercise 01
3-bit RGB color	
Turn-in directory : *ex*01/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	
Notes : `n/a`	

Ok now, more colors! You will have to write a program that controls the RGB LED D5. Here is a table of colors that should be displayed successively, in a loop and changing color every second.

name	R#	G#	B#	color
red	ff	0	0	🟥
green	0	ff	0	🟩
blue	0	0	ff	🟦
yellow	ff	ff	0	🟨
cyan	0	ff	ff	🟦
magenta	ff	0	ff	🟪
white	ff	ff	ff	⬜



	Exercise 02
24-bit RGB color	
Turn-in directory : *ex*02/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	
Notes : `n/a`	

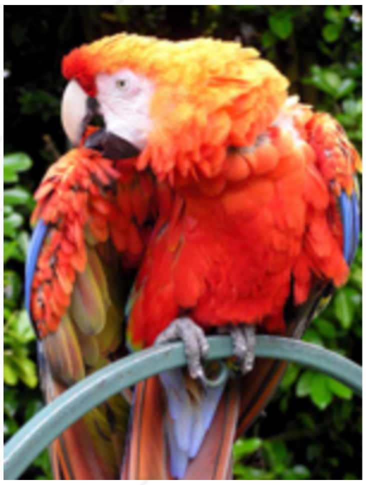
Ok, now even more colors! You must write a program that controls the RGB LED D5 but with PWM. Make the colors transition from one to the next smoothly using the provided wheel function.

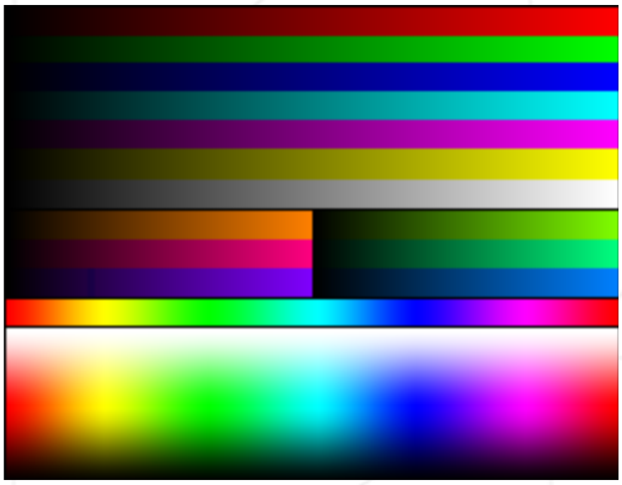
- Write a function init_rgb() that initializes the timers.
- Then write a function set_rgb() that allows you to set the color of the LED.
- Finally, your program must display the color wheel.

`The colors obtained in the requested gradient are always the sum of two primary colors, no more, sometimes less.`

```
void init_rgb();
void set_rgb(uint8_t r, uint8_t g, uint8_t b);

void wheel(uint8_t pos) {
        pos = 255 - pos;
        if (pos < 85) {
                set_rgb(255 - pos * 3, 0, pos * 3);
        } else if (pos < 170) {
                pos = pos - 85;
                set_rgb(0, pos * 3, 255 - pos * 3);
        } else {
                pos = pos - 170;
                set_rgb(pos * 3, 255 - pos * 3, 0);
        }
}
```





Chapter V

Millions of colors

	Exercise 03
UART to RGB	
Turn-in directory : *ex*03/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	
Notes : `n/a`	

You must make a program that listen to the serial port.
When you send a new line with an HEX RGB color with the format `#RRGGBB` sets the color on the RGB LED on pin D5.

`You don't see the exact color on the LED you though you set ? That's normal ! It's possible to correct that problem with a software filter. But it's not mandatory to correct it.`