Institut Puig Castellar

Santa Coloma de Gramenet

Ambilight with Arduino

CFGM Sistemes Microinformàtics i Xarxes

Eric Reche Gil and Joel Olivera Organvidez

SMX2A

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement 3.0 Espanya de Creative Commons

B) GNU Free Documentation License (GNU FDL)

C) Copyright

Resum del projecte (màxim 250 paraules):

Este proyecto tratará de hacer un Ambilight con arduino y montarlo en la pantalla de un portátil.

Un ambilight es una tira de leds que van cambiando de color conforme la imágen que se muestra en pantalla.

Este proyecto innova implementando los LEDS en la pantalla de un portátil, ya que todos los proyectos ambilight se suelen realizar sobre televisores o monitores.

Usando el programa de Arduino y Processing, que es un programa en el que se programa con java, se manda la información de uno a otro, haciendo así que los leds cambien de color dependiendo de la imágen que se muestre por pantalla.

Processing se encontrará continuamente haciendo capturas de pantalla, de las que cogerá el color de x pixeles en x posición, que guardará en un vector de una dimensión, para que luego se envié la información a Arduino por el puerto serie.

Mientras, Arduino estará escuchando por el puerto serie la orden de cambiar x led de x color.

Paraules clau (entre 4 i 8):

- Arduino
- Ambilight
- Leds
- Pantalla/Portátil
- Processing

Índex

1. Introducció

1.1 Context i justificació del Treball

Se ha decidido hacer el **Ambilight** porque como proyecto para utilizar un arduino es una buena idea, además de usar los leds para hacer algo más orientado a lo estético.

1.2 Objectius del Treball

Llistat dels objectius del treball

- Hacer un ambilight funcional
- Crear una página web para el proyecto.

1.3 Enfocament i mètode seguit

Para completar el proyecto, usaremos información de otros proyectos y la adaptaremos para poder utilizar el **Ambilight** en la pantalla de un portátil.
El proyecto está más enfocado a programar que a manipular, ya que tenemos que programar el Ambilight con Arduino y Processing.

1.4 Planificació del projecte

(Diagrama de Gantt)

1.5 Breu sumari de productes obtinguts.

Tenemos que comprar un **arduino** y una tira de **leds WS2812B** (el portatil ya lo teníamos).

1.6 Breu descripció dels altres capítols de la memòria

El resto del proyecto contendrá definiciones, conclusiones, un anexo y en general información sobre lo que era necesario, lo que tenemos en mente para este proyecto y también del montaje y la programación del mismo.

2. Resta de capítols

Ambilight:

El diseño del **Ambilight** será una tira de leds conectados a un **arduino**, pegados a los bordes de la parte trasera de la pantalla. La función del Ambilight consistirá en que los leds tengan la capacidad de cambiar de color dependiendo de la imagen que se muestre en pantalla, de esta forma si ponemos la parte trasera de la pantalla contra una pared, se verán reflejados los colores, siendo simplemente un elemento estético.
La programación del proyecto es en Processing y Arduino, ya que son los dos programas que necesitaremos: Processing para hacer las capturas de pantalla y coger el color RGB de ellas para mandarselas por el puerto serie a Arduino. Y Arduino para leer los datos que le envía processing y cambiar de color los leds, dependiendo de la imagen que se muestre por pantalla.

Material:
Para realizar el proyecto necesitaremos:

- Leds RGB
- **Arduino**
- Portatil/Monitor
- Soldador
- Estaño
- Tubo termoretractil

Valoración económica del proyecto:

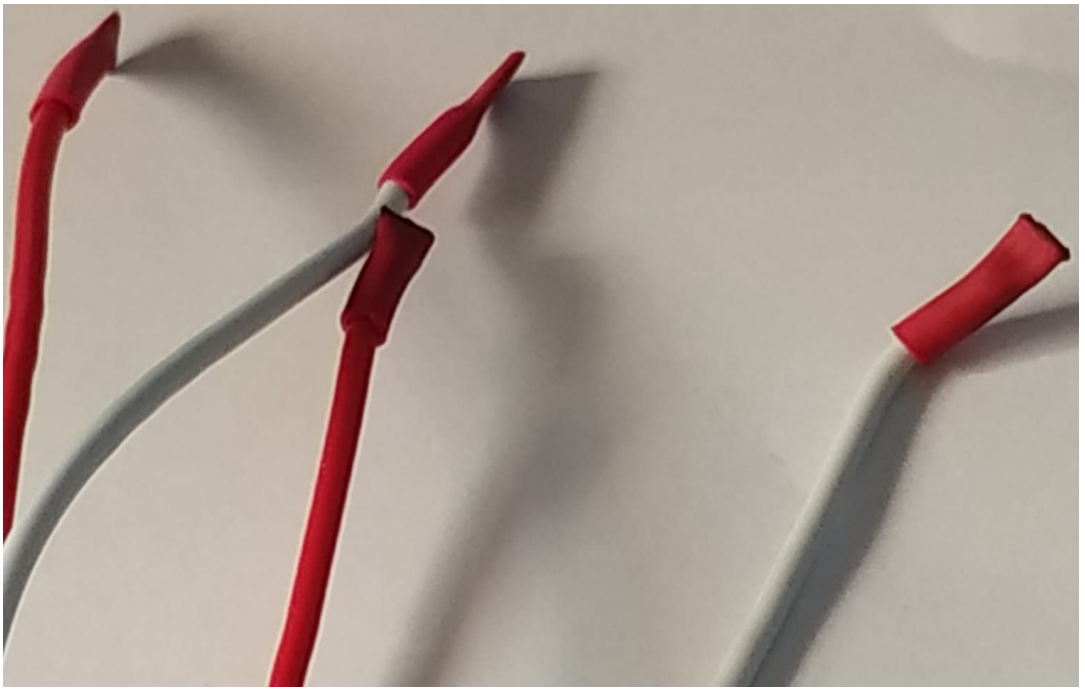
El primer paso será reunir todo el material necesario. Como ya contamos con el soldador y usaremos la pantalla de un portátil, tendremos que comprar por amazon los leds y el **arduino**.
Los leds los hemos comprado por un precio de 31,71€, y el arduino 25,75€.

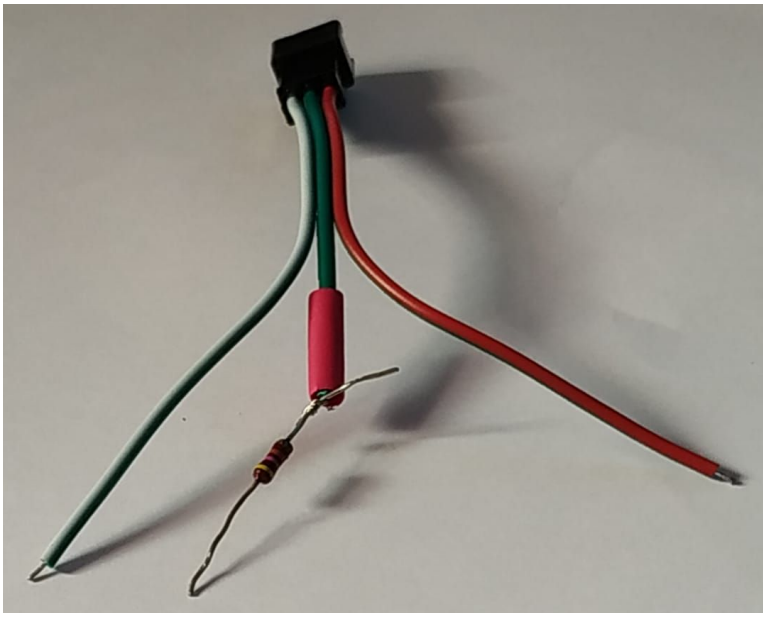
Gastos:

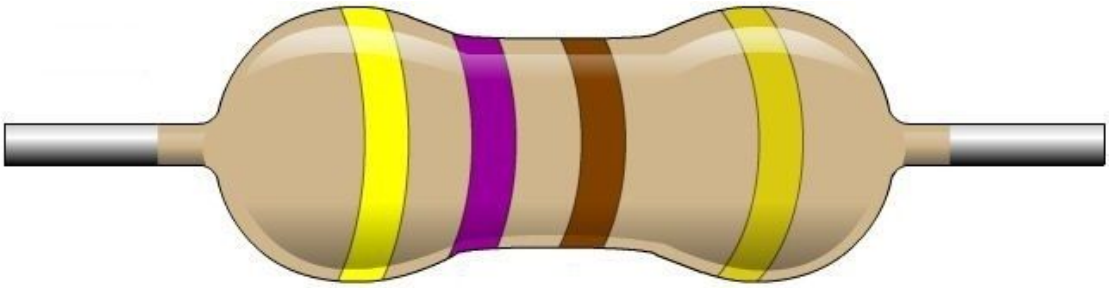
Material	Precio
Leds RGB	31,71€
Arduino	25,75€
Total:	57,46€

Montaje

Para comenzar, sellaremos algunos cables que no sirven con un tubo termoretráctil y soldaremos una resistencia de 470Ω en el cable verde, que es el que irá en el pin 7 del Arduino.



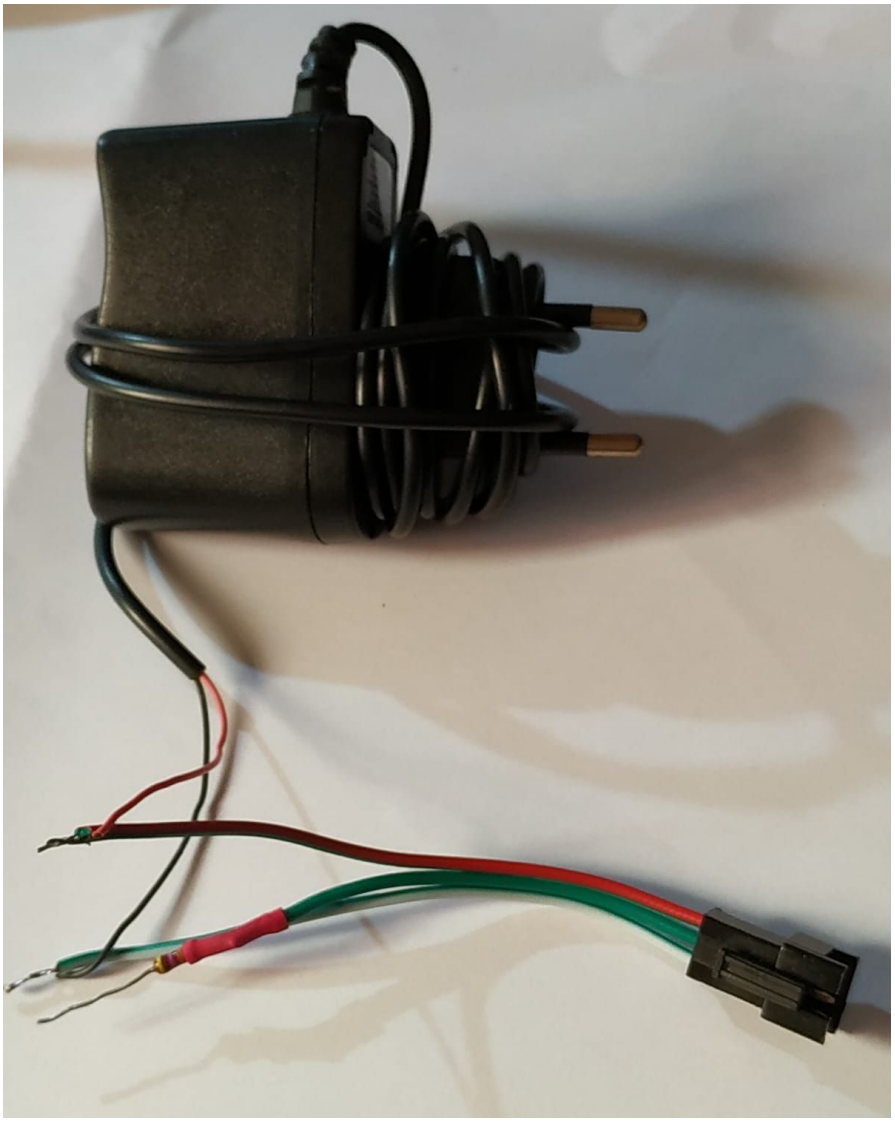




Resistencia 470Ω @ 1/4 watt Tolerancia 5%

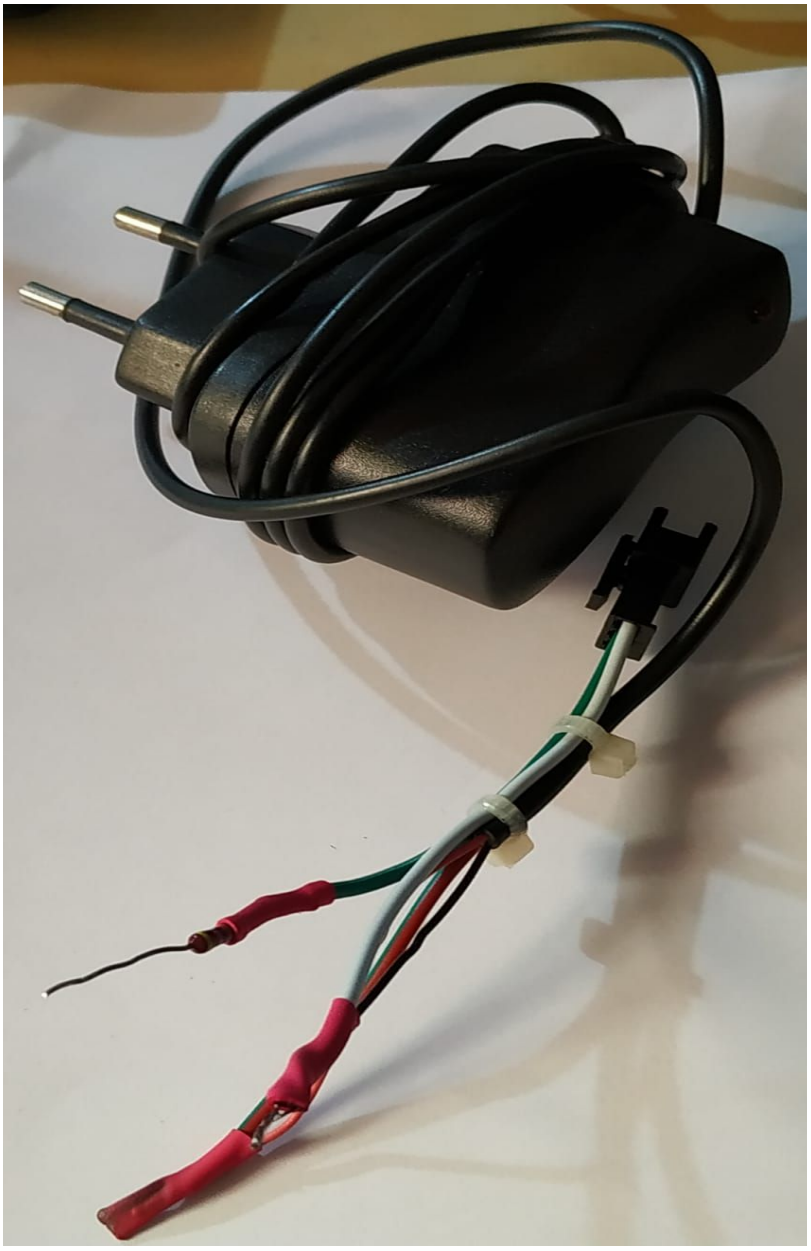
Amarillo	Violeta	Café	Oro

Conectamos el adaptador de corriente a la los leds. El cable rojo del adaptador con el rojo de los leds y el negro del adaptador con el blanco de los leds.

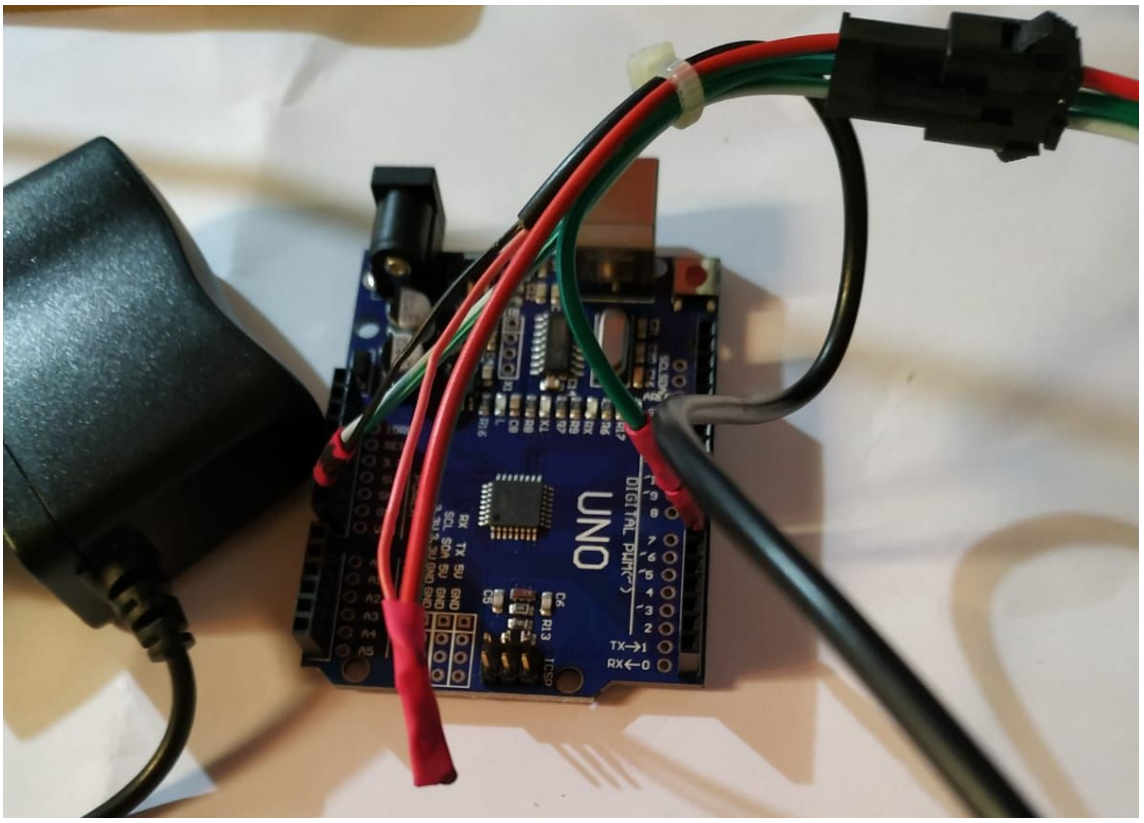


Sellamos la unión del cable rojo del adaptador y el rojo de los leds por la punta, ya que no los vamos a necesitar y también sellamos la unión del negro del adaptador y el blanco de los leds, pero no por la punta, sólo para darle firmeza.

Ahora demostramos cómo sería nuestra explicación y añadimos unas bridas, para que los cables se queden bien juntos y sujetos.



Ahora los conectamos al arduino. El cable verde va al pin 7 y el cable blanco va al GND o “tierra” del arduino.



Como se puede apreciar, el cable blanco tiene una marca negra en la parte del tubo termoretractil sellado, para así diferenciarlo del cable verde y saber instantáneamente cuál conectar y donde.

Al acabar con la anterior tarea, lo siguiente sería probar los leds.
Para realizar este procedimiento, se usará el siguiente código de arduino.
Simplemente se requiere de subirlo a la placa teniendo los leds conectados a la misma.
Aquí adjunto los links de los códigos: link1 link2, aunque abajo he encastado una imágen del mismo abierto con el ide de Arduino.

Link1

```
#include <FastLED.h>
#define LED_PIN     7
#define NUM_LEDS    20
CRGB leds[NUM_LEDS];
void setup() {
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);

}
void loop() {

  leds[0] = CRGB(255, 0, 0);
  FastLED.show();
  delay(500);
  leds[1] = CRGB(0, 255, 0);
  FastLED.show();
  delay(500);
  leds[2] = CRGB(0, 0, 255);
  FastLED.show();
  delay(500);
  leds[5] = CRGB(150, 0, 255);
  FastLED.show();
  delay(500);
  leds[9] = CRGB(255, 200, 20);
  FastLED.show();
  delay(500);
  leds[14] = CRGB(85, 60, 180);
  FastLED.show();
  delay(500);
  leds[19] = CRGB(50, 255, 20);
  FastLED.show();
  delay(500);
}
```

Link2

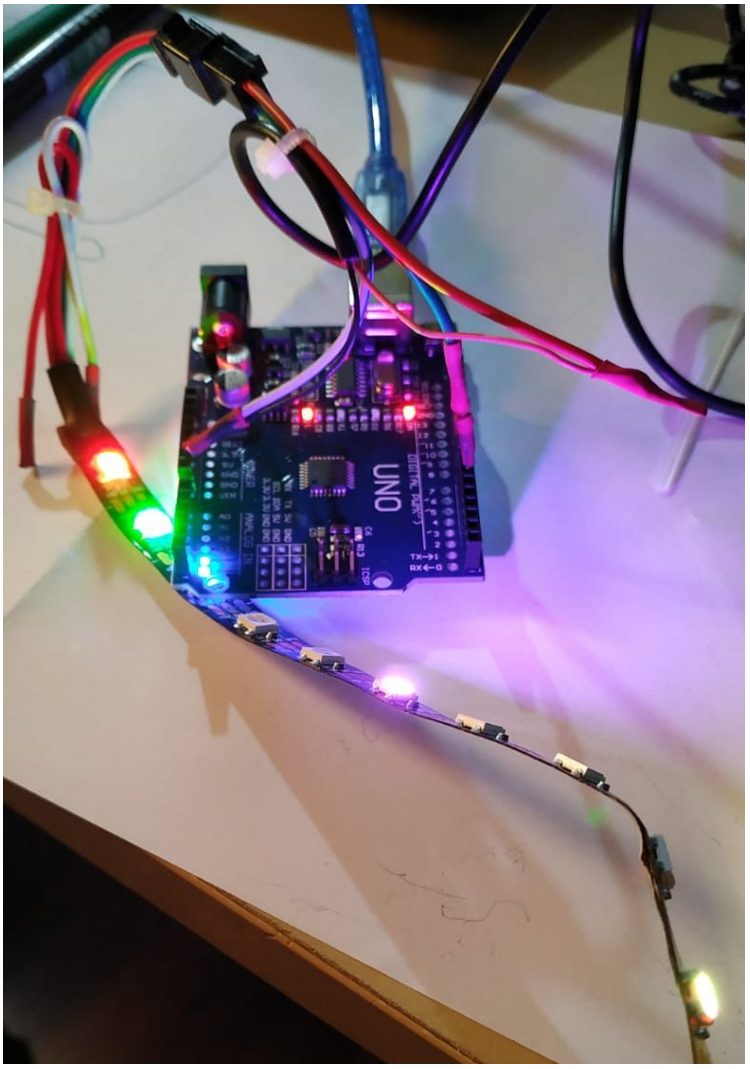
```
#include <FastLED.h>

#define LED_PIN     7
#define NUM_LEDS    60
CRGB leds[NUM_LEDS];

void setup() {
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
}

void loop() {
  for (int i = 0; i <= 60; i++) {
    leds[i] = CRGB ( 255, 0, 0);
    FastLED.show();
    delay(100);
  }
  for (int i = 60; i >= 0; i--) {
    leds[i] = CRGB ( 0, 0, 255);
    FastLED.show();
    delay(100);
  }
}
```

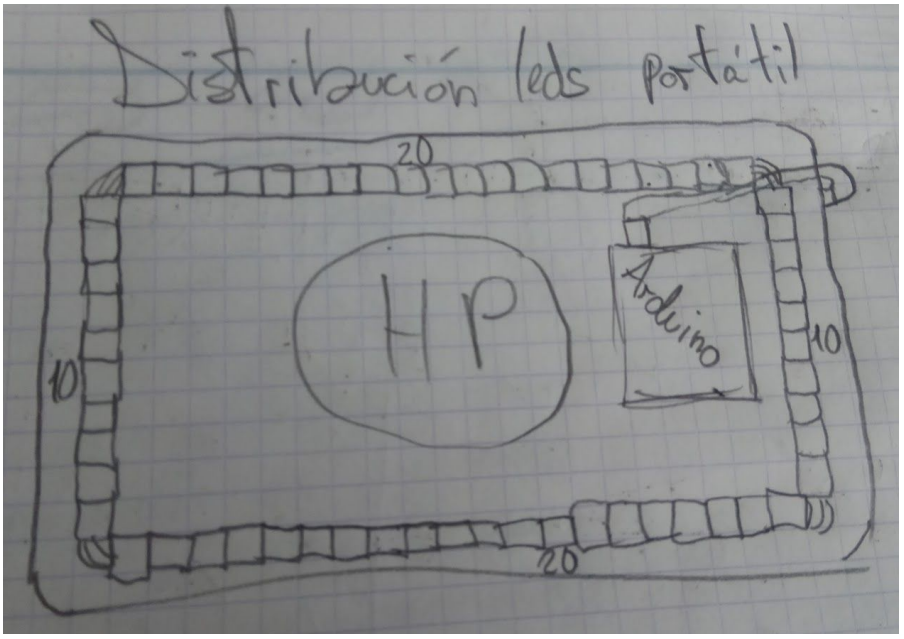
Los dos códigos son válidos para probar el funcionamiento de los leds sin usar, aún, el processing.
En este ejemplo, se usa el código de arduino del link1.
Este sería el resultado del funcionamiento del mismo:



Una vez que comprobado que funciona cada uno de los leds, se realiza un boceto de como quedarían distribuidos los leds por la parte trasera de la pantalla del portátil. Cuando el boceto se acerque a la idea para distribuir los leds, se empieza a cortar por la zona indicada, y se sigue por la parte de cobre, siguiendo dicho esquema.

En nuestro caso, al tener 60 leds, se ha distribuido de esta manera:

- 10 leds en la parte derecha e izquierda de la pantalla
- 20 leds en la parte superior e inferior de la pantalla
- La parte cercana al usb para poder poner el arduino



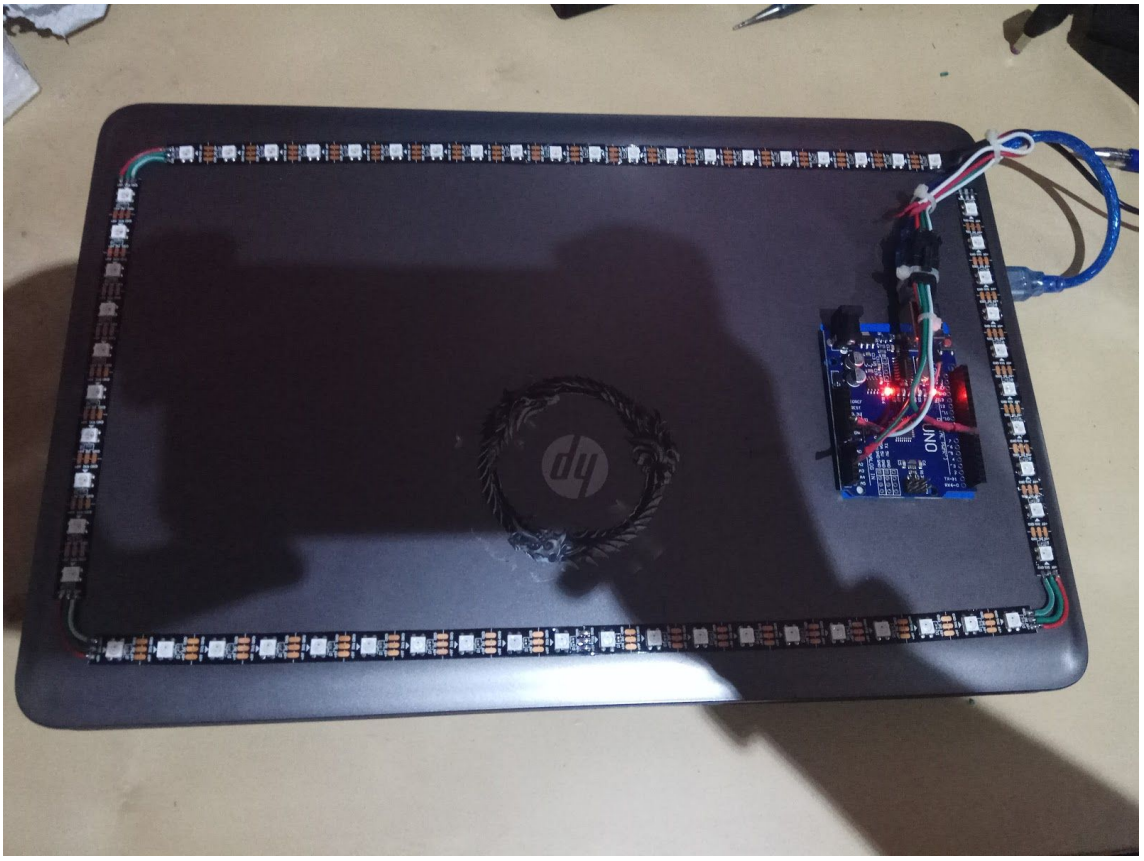
Una vez consolidada la idea y vemos que queda como queríamos, el siguiente paso será soldar los leds.
Los cables que han sobrado de los leds, el blanco, verde y rojo, los vamos a reutilizar para soldar los leds entre sí.
Como se puede apreciar en la siguiente imagen, se han cortado los cables sobrantes a medida y se han soldado correspondientemente a su pin.

Los cables se sueldan en la parte de cobre indicada.
El cable blanco va a la parte que indica que es el GND.
El verde va la parte que indica que es el DO (datos).
El rojo va a la parte que indica que es el +5v.

Una vez se ha terminado de soldar, tendrían que quedar así. (La última parte no puede ir como se ve en la foto, tiene que ir al revés, porque los leds se asignan un número ellos mismos y cuando realizamos la prueba, esta última parte no se encendía. Se desuelda y se coloca como originalmente estaba la série y, por ese mismo motivo, funciona.)

Una vez soldados, con la pega que ya viene en los leds (si no viene, simplemente le aplicamos una cinta de doble cara para pegarlos), finalmente, los pegamos a la pantalla. Nuestra distribución quedaría así:

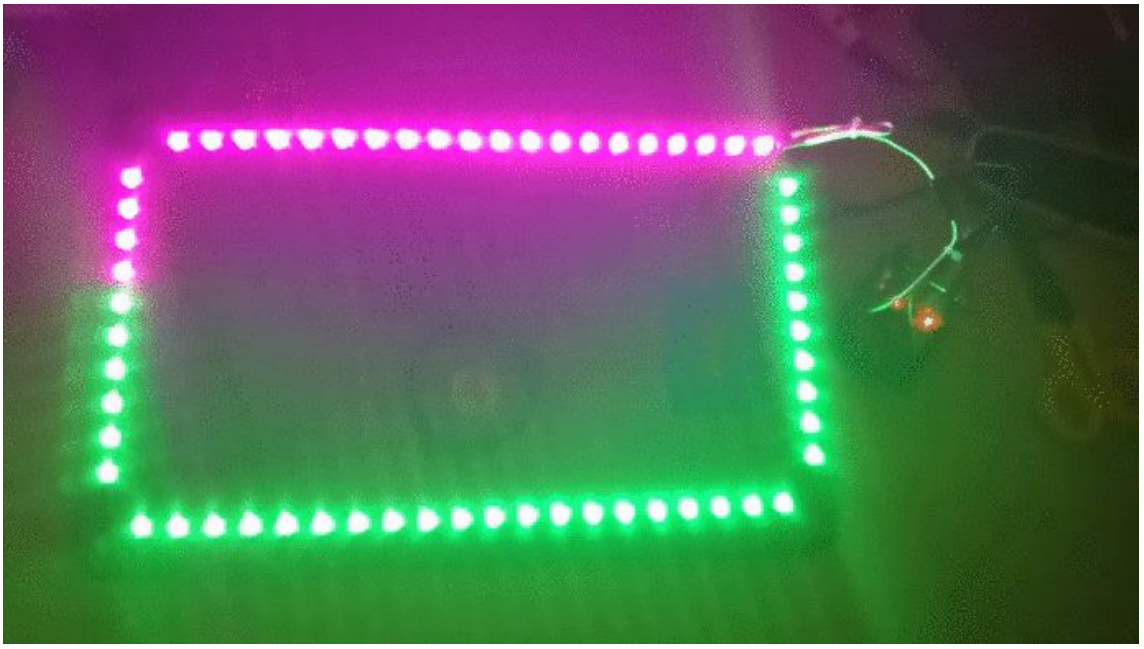
Para definir la zona del arduino, debemos de calcular el largo del cable USB que usaremos para conectarlo al portátil y ponerlo de manera que no moleste a los leds.
Con esto realizado, pegamos el arduino, en la zona marcada con cinta azul, con cinta de doble cara.



El siguiente paso es opcional, pero en nuestro caso lo implementamos para añadir protección a la placa, ya que el proyecto tendría que ser transportar en una mochila, así que, con trozos de goma espuma, le construimos una carcasa al arduino, que también pegamos con cinta de doble cara por la parte de la base a la parte trasera de la pantalla del portátil.

Ahora, como un añadido un poco estético, aunque también da un poco de protección, le añadimos una “tapa” a la protección de goma espuma, que pegamos con cinta de doble cara.

Ahora, es el momento de volver a comprobar el funcionamiento con uno de los dos códigos mostrados inicialmente. Preferiblemente, para mostrar que todos funcionan, es mejor el segundo código.



Pasado un tiempo, los últimos dos meses previos a la entrega del proyecto, tuvimos un incidente con el adaptador de corriente de los leds. Se rompió porque un profesor hizo mucha fuerza al querer enchufarlo a la corriente. Aún así, el adaptador estaba algo deteriorado, así que tuvimos que arreglarlo.

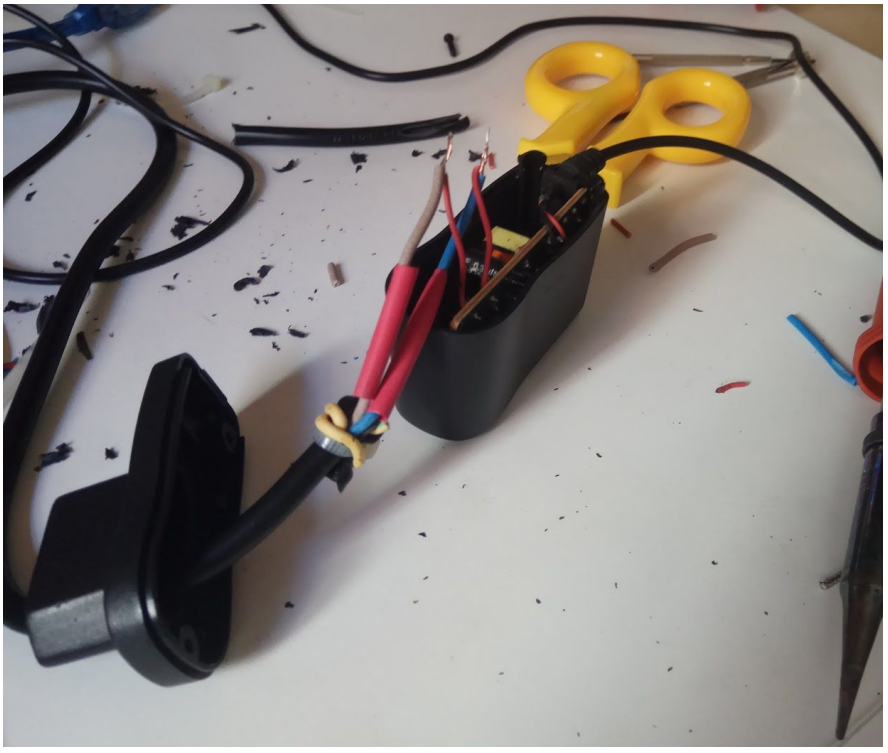
Como se puede apreciar en la imagen, se rompieron las puntas del adaptador de corriente. Así que, para no tener que cortar el cable y crear un empalme, el cual no se sabía de qué forma iba a quedar, por eso se realizó un apaño bastante útil con el que ha quedado muy bien, ya que es mucho más firme que el anterior y nos proporciona más distancia para poderlo enchufar los leds, ya que es más largo.

El procedimiento fue el siguiente:

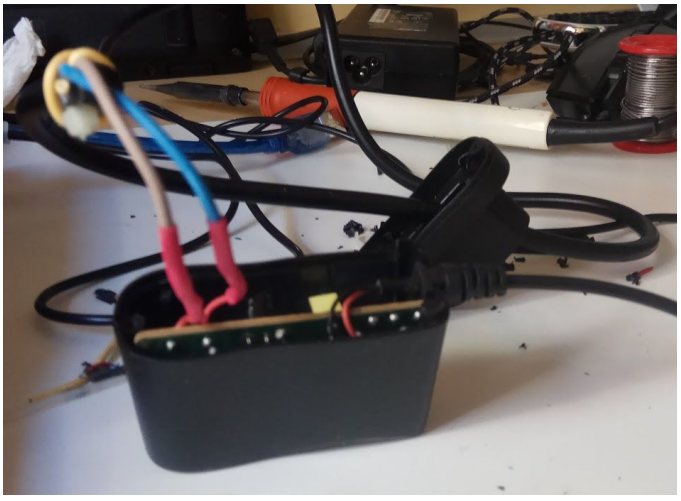
1. Se abre el cargador como se ve en la imágen inferior y le hacemos un agujero con el soldador por donde antes iban las puntas del adaptador.

2. Cogemos un adaptador de corriente de una fuente de alimentación de PC, cortamos la extremidad que no necesitamos y dejamos los dos cables fuera y los pasamos por el agujero que se realizó en el antiguo adaptador.

3. Le hacemos un nudo con el cable que no necesitamos en la parte que va después de la tapa del antiguo adaptador, simplemente para que el cable no se salga y una vez el nudo está bien firme, soldamos los cables del nuevo adaptador con los del antiguo. Para asegurarnos de que la unión soldada no se separe, le ponemos un trozo de tubo termoretractil a cada cable antes de soldar nada.

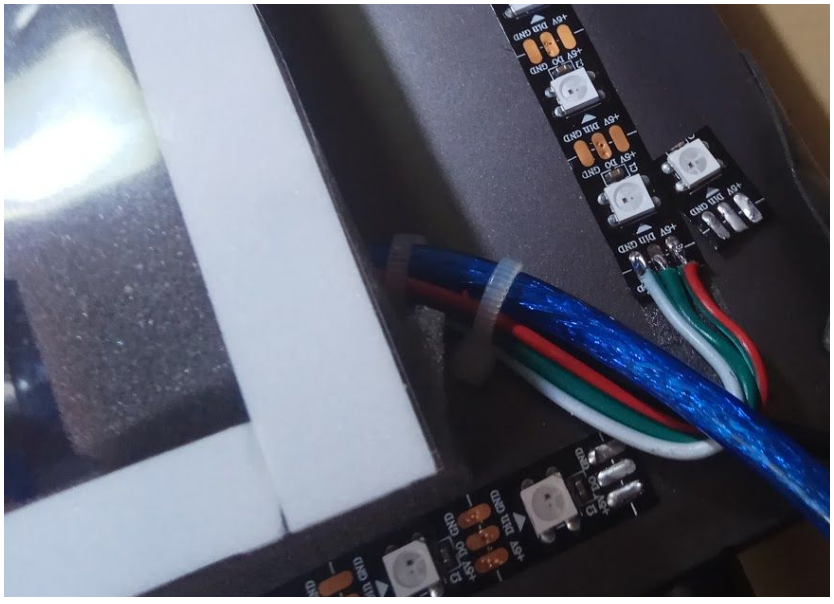


4. En la imagen inferior se pueden ver los cables ya soldados y con el tubo termoretráctil ejerciendo su función. Ahora solo queda cerrar la tapa y atornillarla con el mismo tornillo que tenía, para que así no se salga de ninguna manera.



En la imagen inferior se puede observar el resultado de la operación que le realizamos al adaptador.

Desgraciadamente, en el proceso de restauración del adaptador de corriente, perdimos un led, así que ahora son 59 y no 60 como eran antes.



Programación:

En esta parte de programación del Ambilight, pasaremos a explicar el recorrido que se ha seguido con los programas Arduino y Processing.

Syncthing:

Como este proyecto ha sido realizado por dos personas (Eric y Joel), hemos repartimos las tareas a realizar durante todo el proceso de creación del proyecto. En el momento de compartir documentos como imágenes, archivos o definiciones, utilizamos una herramienta que nos recomendó nuestro profesor Victor Carceler, llamada Syncthing.

Syncthing es una herramienta que nos permite crear un directorio y que otro usuario se conecte al directorio a través de Syncthing para poder almacenar datos o modificarlos.

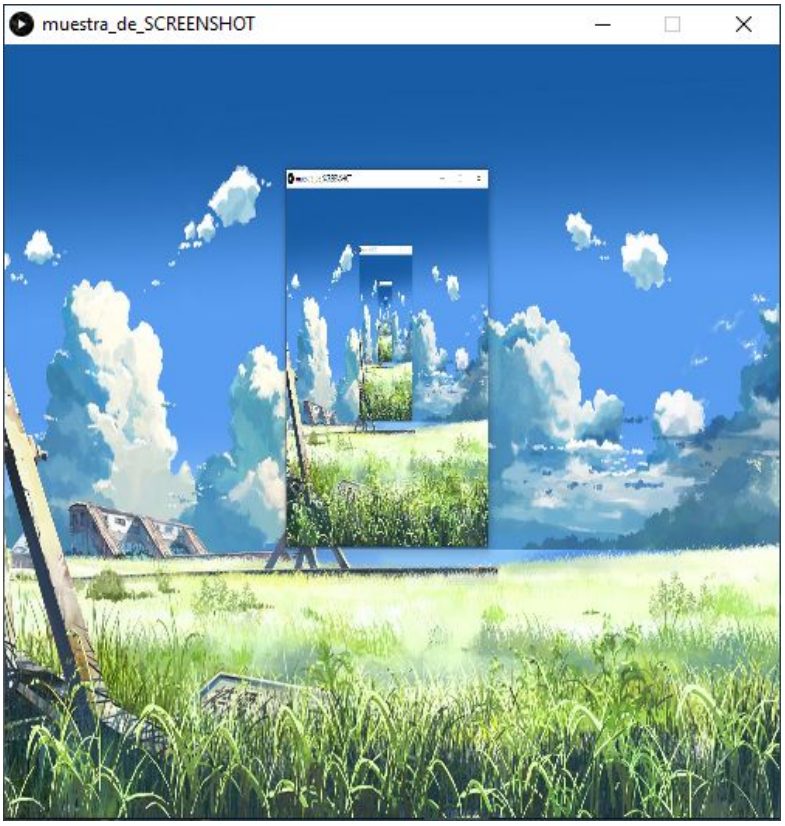
Processing:

El recorrido que se ha realizado con processing ha sido el más largo y difícil, ya que Processing era el programa que tendrá que ser ejecutado para que el ambilight funcione.

Comenzamos con la idea de hacer un bucle de rectángulos que cambiaran de color mientras se hacían capturas de pantalla cada 0.5 segundos y que de estas, se cogiera el color de x pixel en x posición y dependiendo del color del pixel, el rectángulo cambiaba de color.

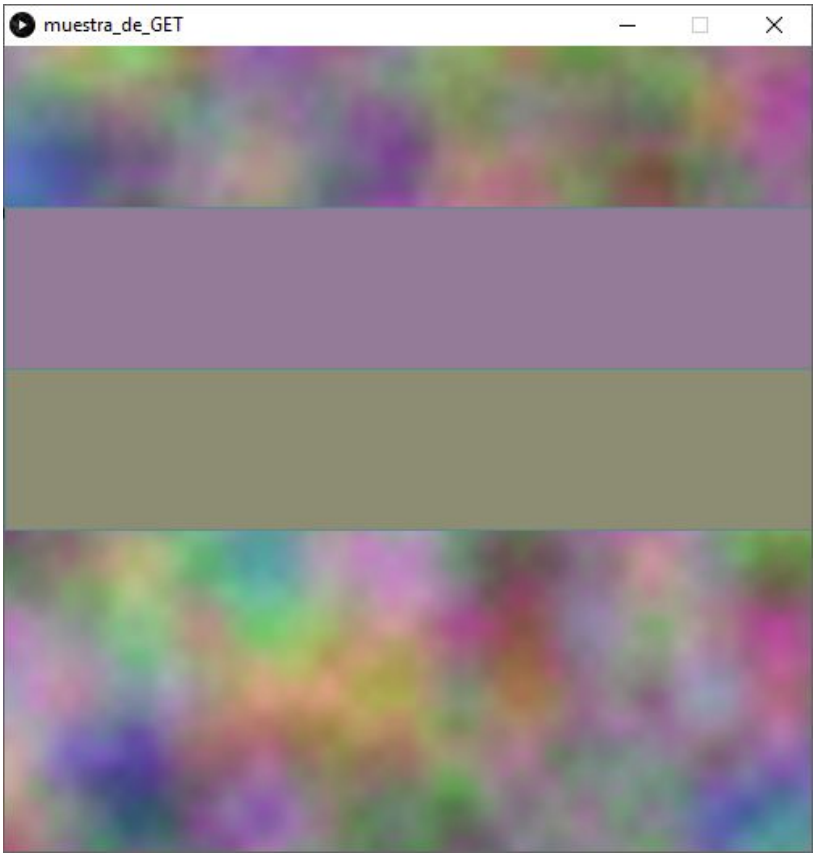
Los tres primeros ejemplos que realizamos fueron los siguientes:

1. Hacer capturas de pantalla



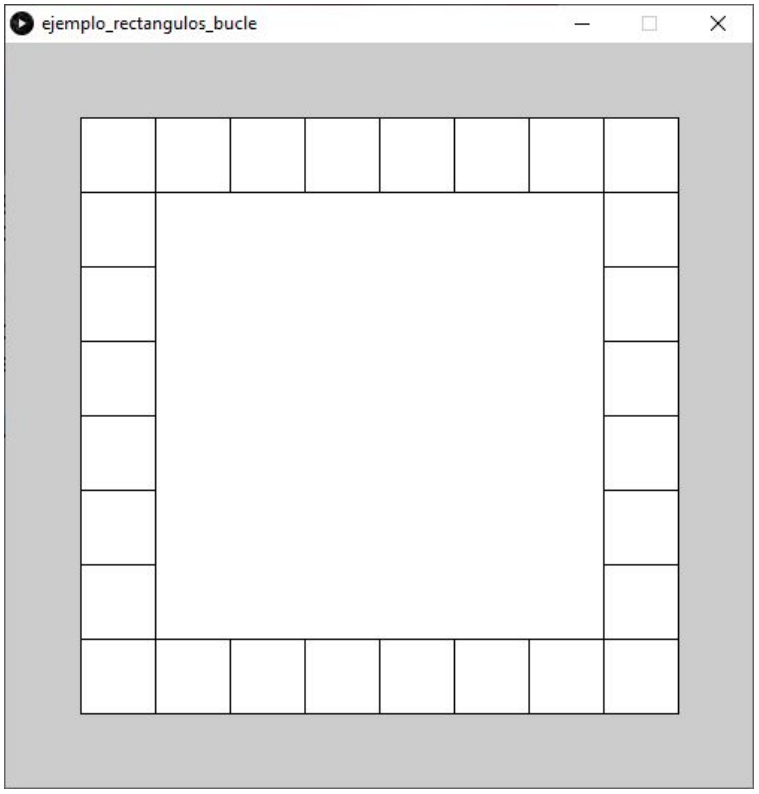
Como se aprecia en la imágen del programa ejecutado, va haciendo capturas y las va mostrando.

2. Coger color de x pixel en x posición



Desgraciadamente, en esta no se puede apreciar su funcionamiento. Lo que hace es coger el color sobre el que está puntero del ratón (únicamente funciona en la neblina de colores que se muestran dentro de la pantalla ejecutada) y en función del color recogido por la función get, el rectángulo superior va mostrando tal color.

3. Bucle de rectángulos

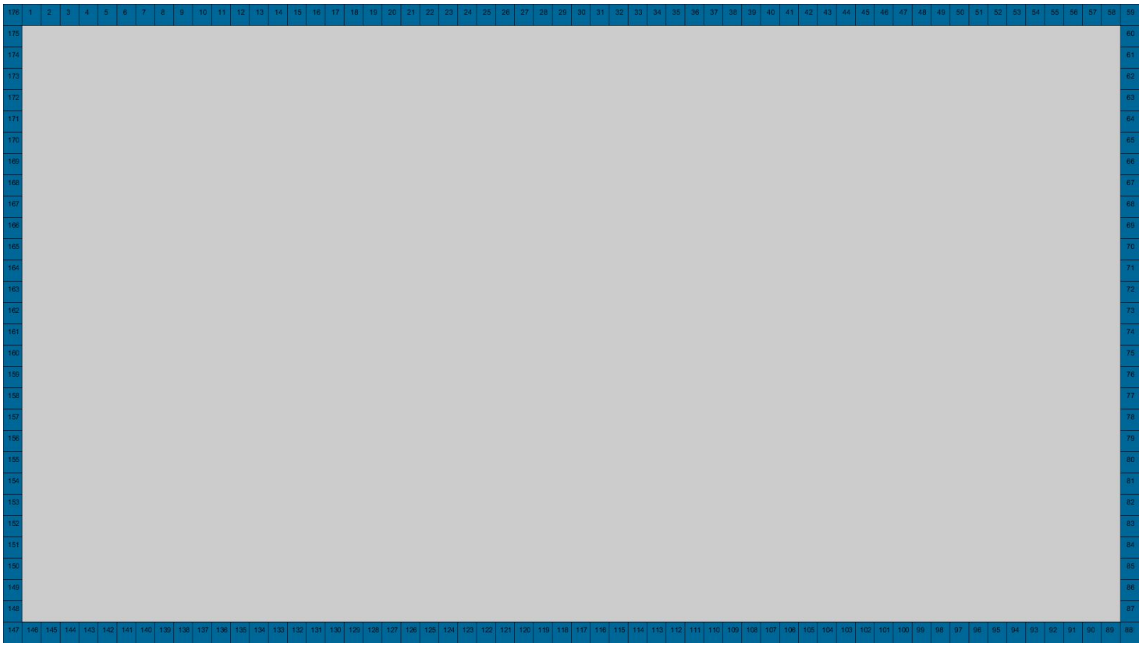


Como se puede apreciar perfectamente en la imágen superior, al ejecutar este programa se crea un bucle de cuadrados dentro de otro cuadrado haciendo un marco en el mismo.

Poseyendo estos tres ejemplos, damos paso a la unión de los mismos, para así implementar nuestra idea principal, hacer un bucle que pinte cuadrados en forma de marco dentro de la ventana del ejecutable, hacer capturas y coger el color x pixel en x de estas capturas, para que los cuadrados se pinten en función del color recogido.

4. Primer bucle

Este fue nuestro primer bucle de cuadrados que formaba un rectángulo. Este ejemplo no tenía ni pies ni cabeza, ya que habia cuadrados que se salían y la numeración de los mismo era errónea.

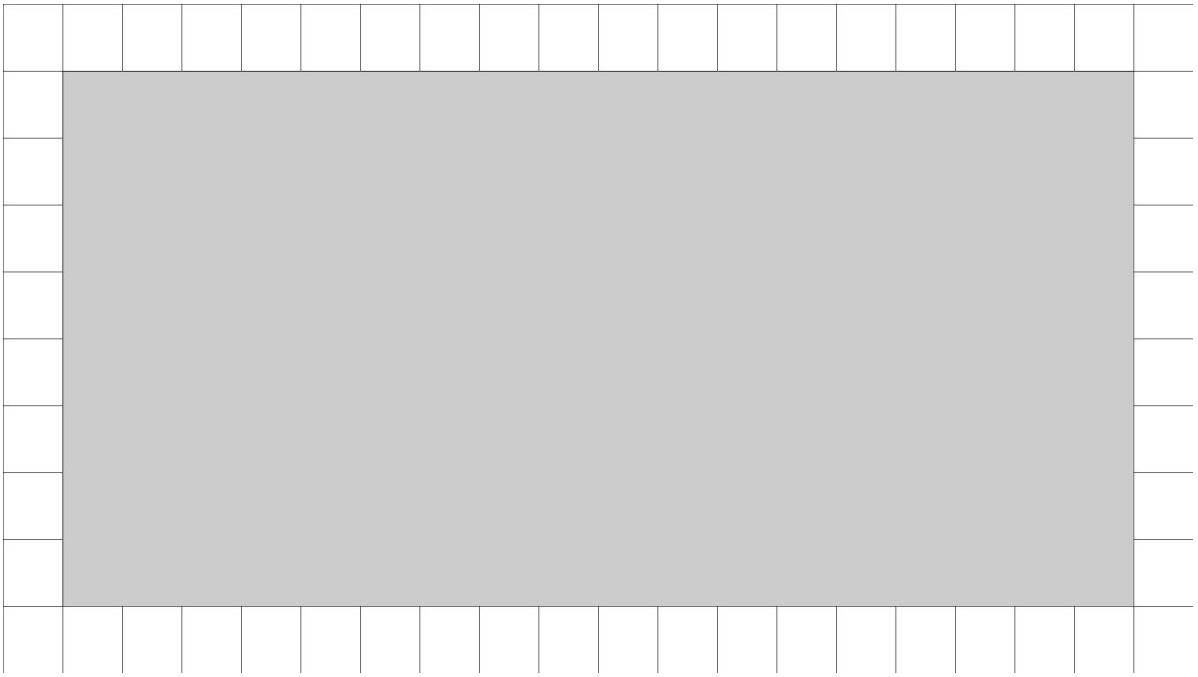


La imagen superior muestra el ejemplo del primer bucle, pero no se puede apreciar el orden de los cuadrados porque la imagen está reducida para que quepa en el documento.
Como se puede observar en la imágen, hay un marco de cuadrados azules, unos más grandes que otros, que tienen asignado un número en el centro del mismo cuadrado.

Abandonamos este ejemplo, ya que no funcionaba como se esperaba, así que comenzamos uno nuevo y este si que se parecía mucho más a lo que buscábamos.

5. Coger color cuadrados y pintarlos

Este ejemplo era más complejo que el anterior y funcionaba correctamente, pero en nuestro caso, hubo un inconveniente que nos hizo replantearnos la idea de pintar los cuadrados.



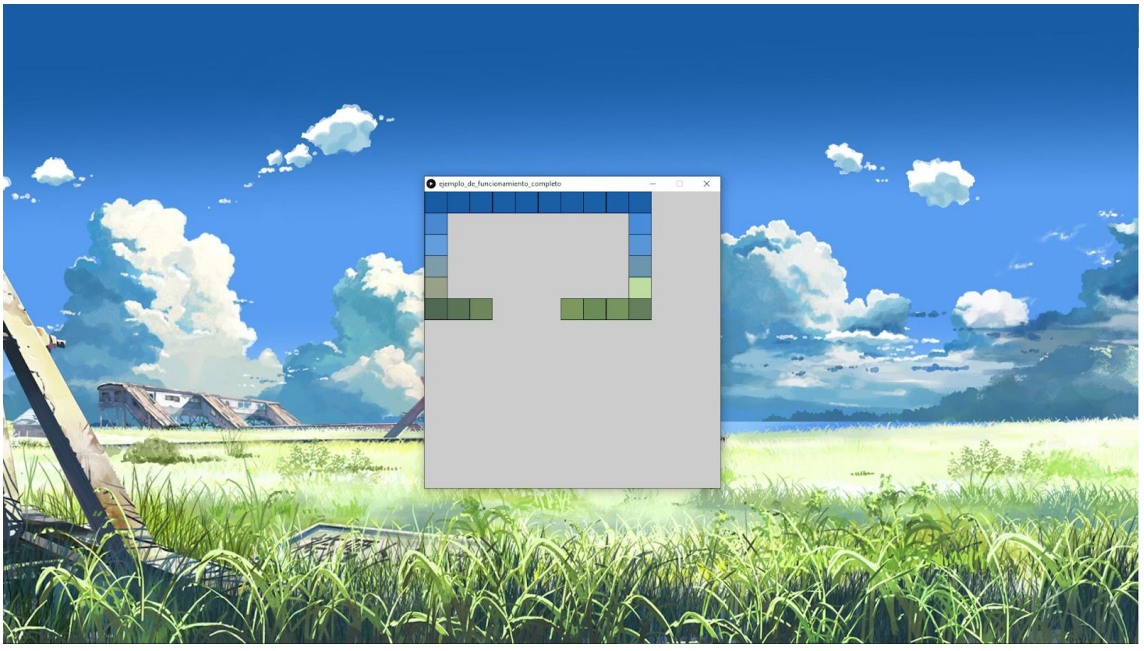
La imagen superior se ve de pena, pero pinta los cuadrados que son (20x10).

Con este ejemplo queríamos coger el color de las capturas de dentro del cuadrado y pintar este mismo del color recogido. Pero el color lo cogía del mismo cuadrado y no de las capturas, entonces se decidió descartar la idea ya que los cuadrados sólo eran un obstáculo en nuestro código.

6. Shapes
Shapes fue una recomendación que nos comentó nuestro profesor Jordi Farrero, ya que el programa anterior no era funcional porque cogía el color de los cuadrados pintados en blanco, y lo único que hacían era seguir pintándose en blanco.
Después de haber ojeado la documentación, decidimos que buscar otra idea para continuar con la programación de Processing sería lo mejor.

7. Ejemplo pintar cuadrados

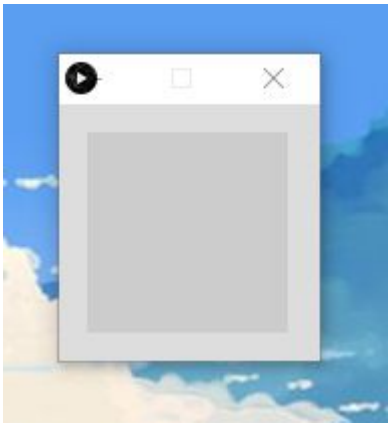
Este ejemplo se parece mucho a la idea que se necesitaba para el proyecto.
Grácias a este ejemplo, desarrollamos la siguiente idea, con la que ya casi finalizaríamos con processing.



Como se ve en la imágen superior, el programa captura el color de x píxeles y hace que los cuadrados cambien de color en función del mismo.

8. Conectar Arduino y processing

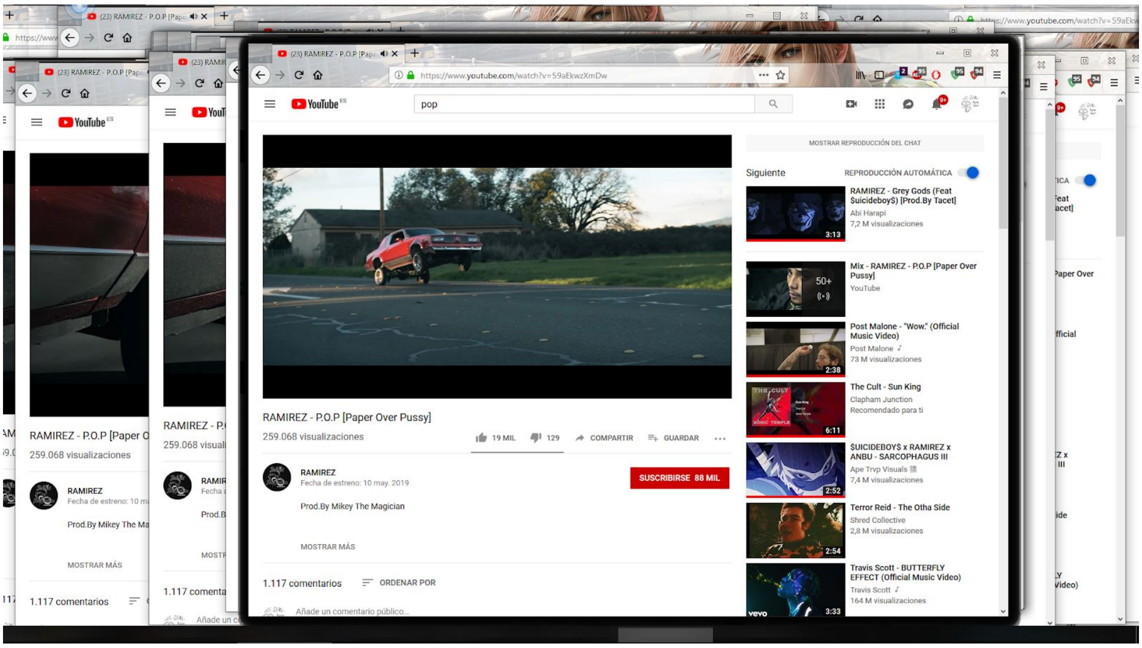
Esta parte consta de un ejemplo muy simple. Este ejemplo se usa para ver el funcionamiento de conectar arduino y processing.
Aquí podemos observar que processing hablaba con arduino por el puerto serie, enviándole datos por el mismo.

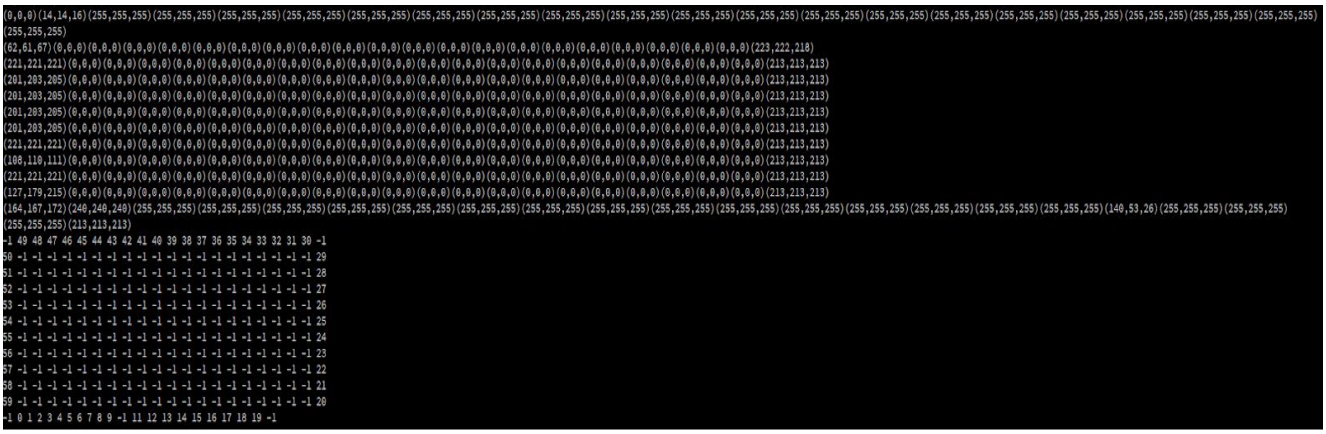


En la imágen superior no se puede apreciar, pero el funcionamiento de este ejemplo es muy básico.
Cada vez que haces clic dentro de la pantalla ejecutada, Processing recibe la orden de enviarle un 1 a Arduino y mientras no haces clic dentro de dicha ventana, le manda 0.
En arduino, se podría poner que cada vez que reciba el número 1 enviado por processing, se encendiera un led y que cada vez que recibiera el número 0, se apague.

9. [Cogiendo color de capturas con arrays]()

Aprendiendo de los anteriores ejemplos, se realizaron dos arrays. En uno se definirá el color del píxel y en el otro el identificador o numerador del led.





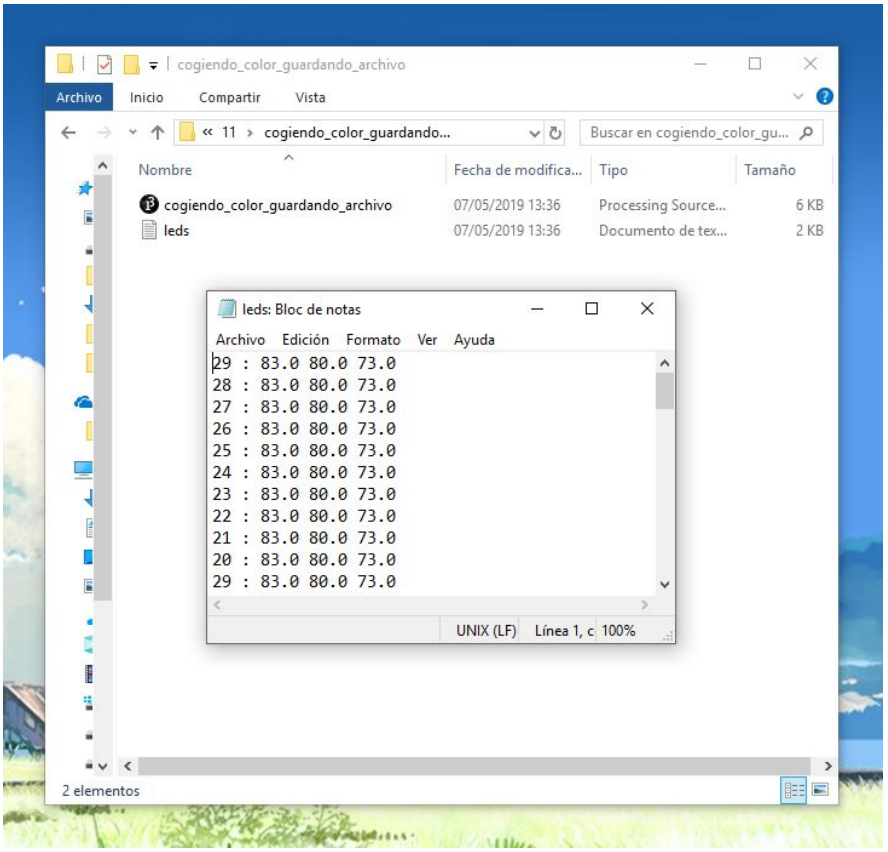
En la primera imágen superior, podemos observar como el programa toma capturas de pantalla y en la segunda, aunque no se pueda llegar a entender lo que muestra, muestra la array con el color recogido y debajo de esta, otra array con el número asignado de cada posición.

10. Cogiendo color capturas más arduino
Con este ejemplo, simplemente se implementa la parte de enviarle los datos al arduino del ejemplo anterior.
Este ejemplo tendría que haber sido el definitivo, pero para nuestra sorpresa, descubrimos que cuando le enviaba el número identificador del led y el número de los colores RGB recogidos de la captura, la información que le llegaba la placa arduino se perdía, haciendo que los leds se pinten de colores aleatorios, ya que recibía datos corruptos que se interpretaban a medida que llegaban.

Al realizar el anterior ejemplo no funcionó correctamente, ya que se perdió la información enviada desde Processing a Arduino a través del puerto serie. Pero,gracias a un compañero, conseguimos realizar el siguiente ejemplo:

11. Cogiendo color y guardando archivo

Como bien se ha explicado antes, es el mismo ejemplo, al que únicamente le ordenamos que guarde en un archivo lo que se envía al Arduino.



Como se puede apreciar en la imagen superior, Processing guardaba el número identificador del led y los valores RGB en un archivo que él mismo creaba.

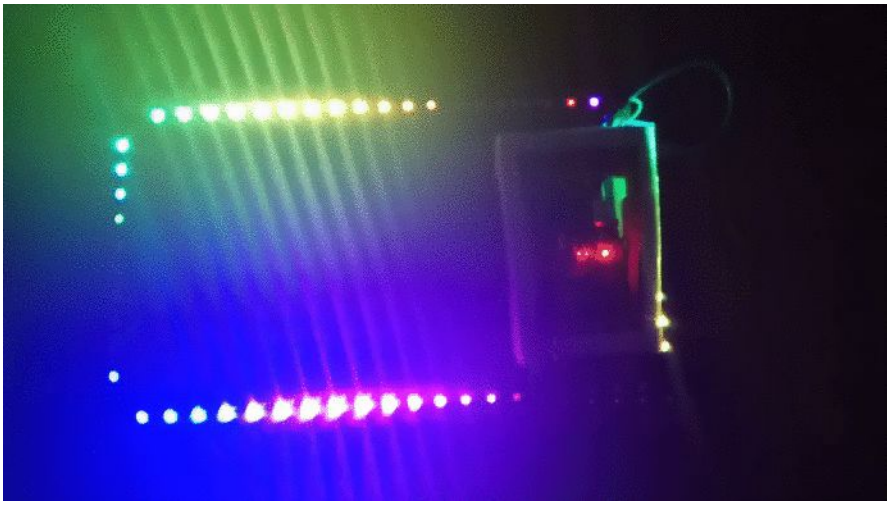
Tras estar trabajando con Processing, comenzamos a buscar proyectos de otras personas para poder realizar el proyecto sin tantos errores o implementar mejoras.

Entonces descubrimos los proyectos de dos usuarios que funcionaban correctamente, permitiendo almacenar los datos en un buffer y enviarselos a Arduino sin perder nada de información.

12. Leds gaming

El nombre del ejemplo número 12 se debe a que, al tener el programa subido en el arduino y ejecutar el ejemplo de processing, los leds se encienden formando una ruleta de todos los colores visibles por el ojo humano.

Fue un resultado muy inesperado pero, gracias al mismo, retomamos energías para continuar con el proyecto.



Como se ve en la imágen superior, los leds hacen el procedimiento que describiamos, se encienden formando una ruleta de colores.

13. Cogiendo color más buffer

Con este ejemplo final de Processing, usamos el anterior ejemplo y lo adaptamos a nuestras necesidades.

El anterior ejemplo almacenaba los datos en un buffer y luego, una vez completo el buffer, lo enviaba a través del puerto serie, evitando así la pérdida de datos.

Como se ve en las imágenes superiores, el programa hace, por fin, la idea que teníamos planeada.

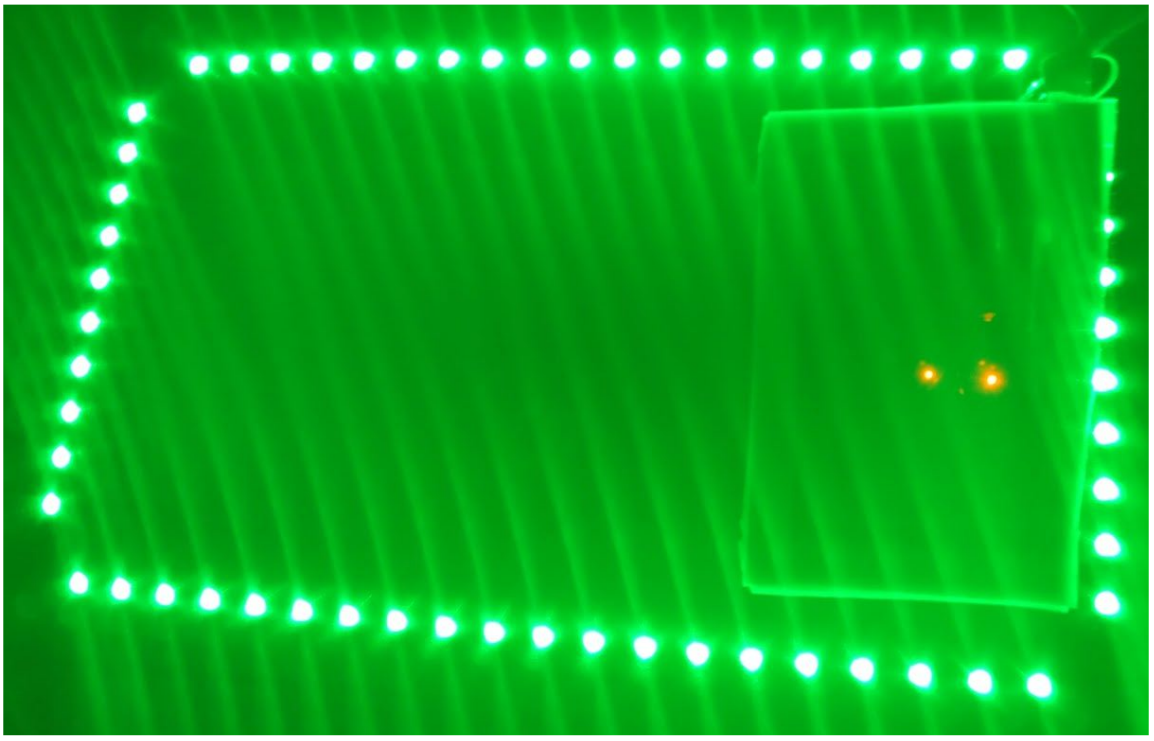
Lo que es sorprendente es que este no utilizaba el void draw, si no un for (;;) que funciona como un bucle reemplazando al draw. Al no tener draw, las capturas no se muestran por pantalla, pero aún así tenemos que tener la ventana gris abierta y poner el video encima de ella para que funcione.

Este ejemplo hace capturas, de las cuales coge el color y, por la posición definida en el vector de una dimensión, va almacenando el número identificador del led y los valores de color RGB en el buffer, para posteriormente enviarlo al Arduino y que este mismo lo interprete y vaya pintando los leds.

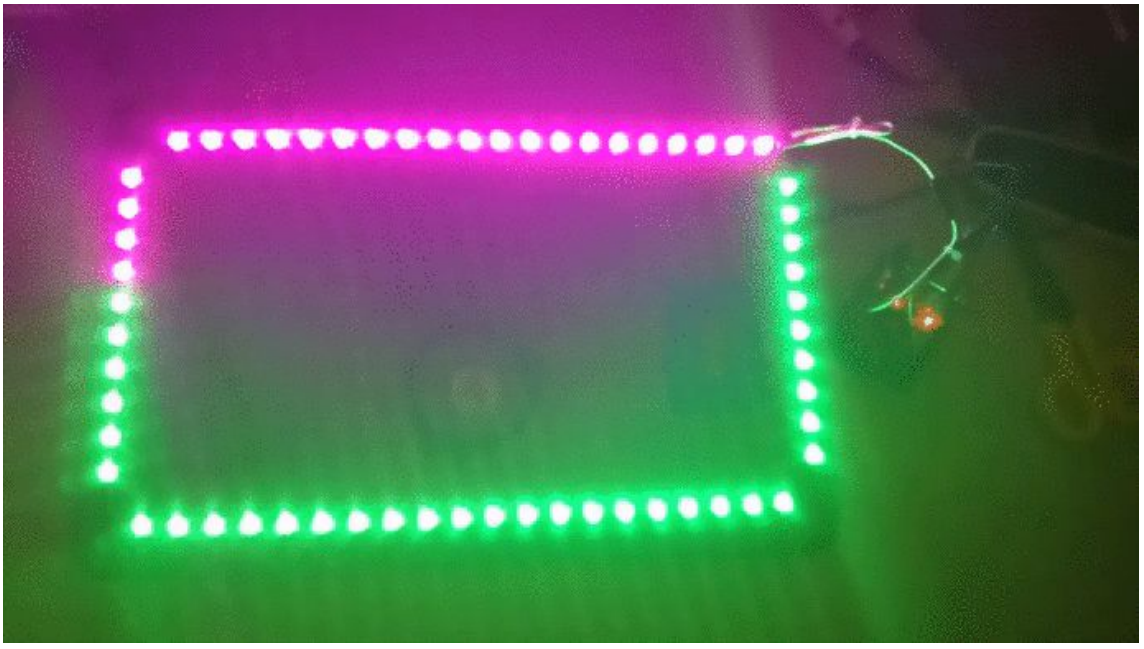
2.3**Arduino:**

Este proyecto no se ha centrado tanto en Arduino, ya que este se basa más en las funciones de processing, aunque Arduino también era necesario para poder controlar la placa, así que también se ha trabajado algo en el funcionamiento de la placa.

1. Ejemplos 0a y 0b
El ejemplo 0a hace que se enciendan los leds que le digas y del color que le digas. En la imagen inferior se puede apreciar el funcionamiento:



El ejemplo 0b hace que se enciendan todos los leds indicados y haga vuelta y vuelta de un color u otro usando bucles. Con la imagen inferior se entiende mejor el resultado del funcionamiento.



2. Primer ejemplo conectar arduino y processing

Este ejemplo de Arduino lo usamos para ver el funcionamiento de la entrada de datos por el puerto serie enviados desde otro programa a Arduino.
El primer problema con el que encontramos al ejecutar los dos códigos a la vez, el de arduino y processing, y querer mirar el monitor serie para ver la llegada de datos a Arduino, era que el puerto serie estaba ocupado y no permitía realizar varias acciones al mismo tiempo. Únicamente podía o bien abrir el puerto serie o bien leer los datos.
El ejemplo recibe x valores y dependiendo de los mismos, se enciende o se apaga el led que va al pin 13 de la placa Arduino.

4. Leds definidos uno a uno

Este fue una idea que se pensó gracias al ejemplo 0a, pero que al final no era lo que necesitábamos.
La idea constaba de tener definidos el id de los leds y qué processing simplemente tuviera que enviar los valores de color RGB a cada led.

5. Intento de usar Adafruit

En este ejemplo buscamos por una nueva manera de hacer funcionar los leds. En vez de usar la librería FastLed como hicimos en el resto de ejemplos, en este usamos la librería de Adafruit Neopixel, que es ligeramente diferente.
Este código lo que pretende hacer es cambiar de color los leds dependiendo de los datos recibidos y pintar "y" por el serial.
Al final se decidió no seguir con esta librería, ya que se usó durante toda la fase de desarrollo de proyecto las librerías de Fast Led, así que abandonamos este ejemplo.

6. Leer fichero processing

Como ya se ha mencionado en processing, este ejemplo consiste en que processing exporte un archivo para que luego arduino lo leyera. Este era el ejemplo de Arduino que se realizó para que leyera el código de dicho archivo.
En el código se puede apreciar cómo se introdujo la localización del archivo exportado por processing y se le ordena que guarde la información en variables, para luego pasársela a los leds.
No fue posible usar este código, porque Arduino no podía leer el archivo ya que no podía guardar los datos en memoria, y se necesitaría una SD para ello.

7. Llegada y pérdida de datos

Con este código conseguimos de una vez por todas que los leds funcionasen, ya que recibían los datos que Processing enviaba, hasta que se descubrió que se perdía la información por el puerto serie y pintaba los led de cualquier color. Cuando ocurrió esto, no quedaba ninguna idea.
El código leía los datos que le llegaban y los clasificaba como el id del led y los valores RGB, pero como se perdían datos en el camino, le enviaba a la placa datos erróneos y por eso pintaba lo que quería, porque si no llegaba un dato, cogía el siguiente.

Buscando por internet más proyectos, encontramos un código de arduino que corregimos y adaptamos a nuestras necesidades.

8. Código Arduino definitivo

Como ya se ha mencionado antes, este es un código que fue usado a partir de un proyecto de internet y lo adaptamos para que funcionara correctamente con nuestro Processing.
Este ejemplo es perfecto para implementarlo en nuestro proyecto, ya que espera a que le llegue el buffer completo y apaga los leds al cabo de x segundos si no le llega información a través del puerto série.
Cuando comprueba que el buffer se encuentra completo, hace una checksum y si esta es correcta, le manda la información para que arduino pinte los leds del color indicado.
También comprueba si el buffer no está completado antes de enviar los datos.

3. Conclusions

- Una descripció de les conclusions del projecte: Quines lliçons s'han après del projecte?.

  Con este proyecto hemos aprendido a soldar con un soldador y estaño y a cómo controlar los leds usando diferentes programas, a programarlos y a hacer que se comuniquen entre los mismos.
  También hemos aprendido que es importante administrar bien el tiempo a la hora de hacer el proyecto, ya que sin haberlo hecho, nos habríamos quedado cortos.

- Una reflexió crítica sobre l'assoliment dels objectius plantejats inicialment: Hem assolit tots els objectius? Si la resposta és negativa, per quin motiu?

  Hemos realizado todos los objetivos planteados inicialmente, ya finalmente decidimos que usar el portátil sería la opción más sencilla y económica. Aunque no los que nos han ido surgiendo a lo largo del proyecto, ya que eran mucho más complicados de hacer y no quedaba tiempo suficiente como para cumplirlos.

- Una anàlisi crítica del seguiment de la planificació i metodologia al llarg del producte: S'ha seguit la planificació? La metodología prevista ha estat prou adequada? Ha calgut introduir canvis per garantir l'èxit del projecte? Per què?

  Hemos intentado seguir la planificación, aunque en algún nos hemos relajado un poco, lo hemos conseguido recuperar en el tiempo estimado. La metodología del proyecto ha sido la misma durante todo el proyecto y creemos que ha sido la acertada, ya que si no, no podríamos haber acabado el proyecto dentro del plazo de tiempo que teníamos para entregarlo.

- Les línies de treball futur que no s'han pogut explorar en aquest projecte i han quedat pendents.

  En un futuro nos gustaría poder controlar los leds desde una página web/bluethoot, hacer que el processing se ejecute en segundo plano, gastando así menos recursos del ordenador y permitiendo usarlo mientras tenemos abierto otro proceso que requiere de más recursos. También nos habría gustado haberle puesto una protección a los leds, ya que estando al descubierto, seguramente se acaben dañando.

4. Glossari

Arduino: Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

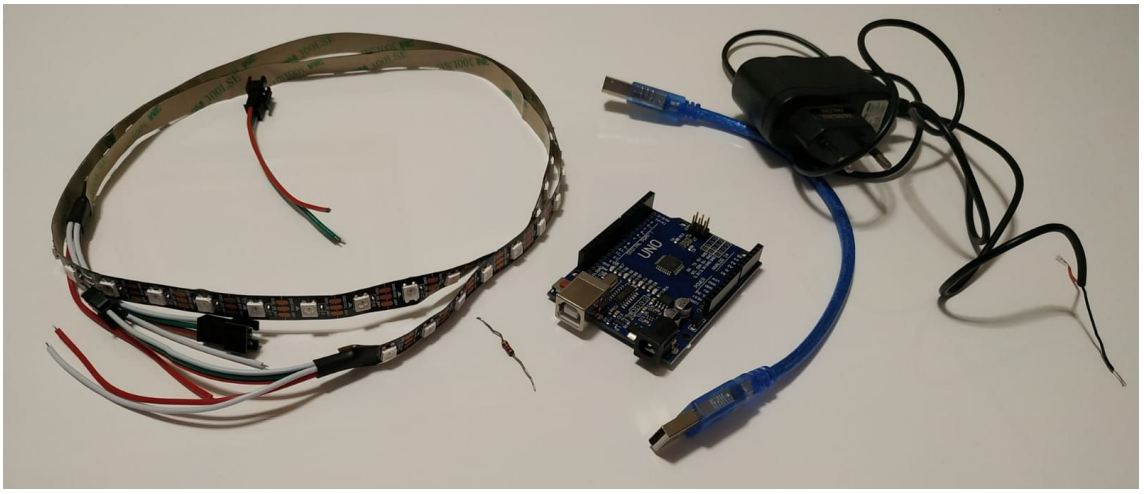
Ambilight: Ambilight es una tecnología diseñada para mejorar la experiencia visual. Sea cual sea la fuente de la señal, la tecnología Ambilight analiza las señales entrantes y produce la luz lateral ambiental adecuada para el contenido que se está visualizando en la pantalla.

Leds WS2812B: Los WS2812B son LED que disponen de lógica integrada, por lo que es posible variar el color de cada LED de forma individual (a diferencia de las tiras RGB convencionales en las que todos los LED cambian de color de forma simultánea).

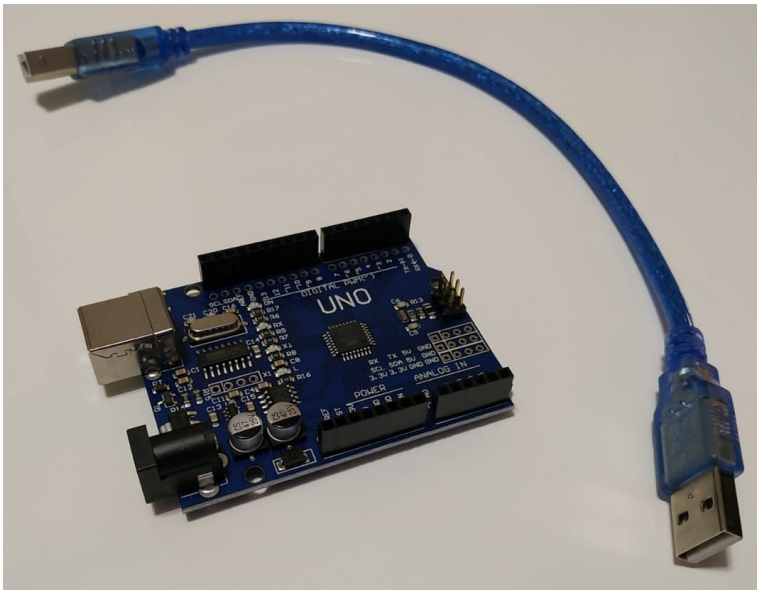
Processing: Processing es un programa de software flexible y un lenguaje para aprender a codificar en el contexto de las artes visuales. Nos permite programar en java, javascript, python, android y raspberry pi.

5. Galería

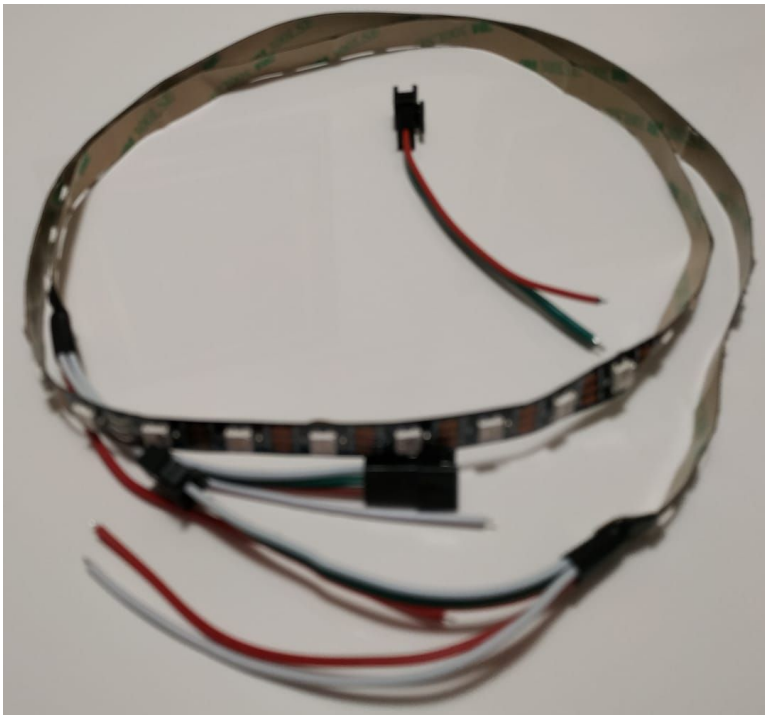
Material:



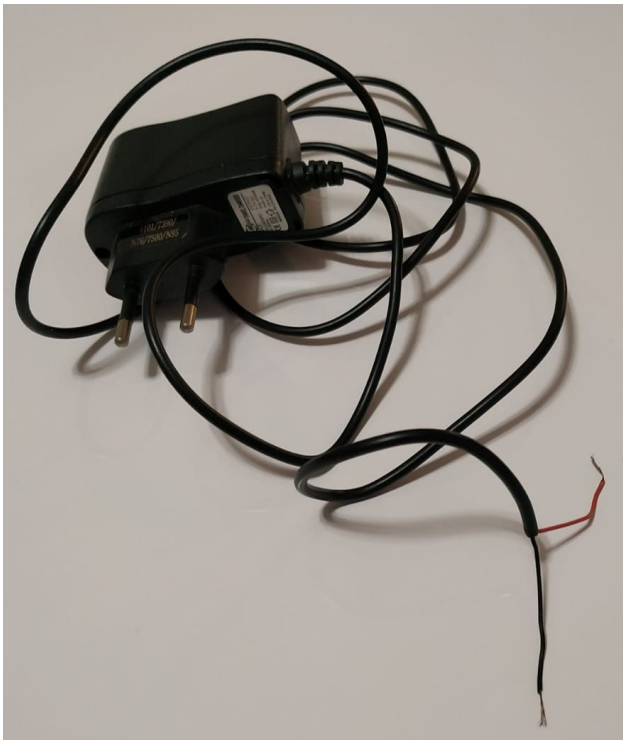
Esta es una foto de todo el material que usaremos



Arduino



Tira de leds ws2812b

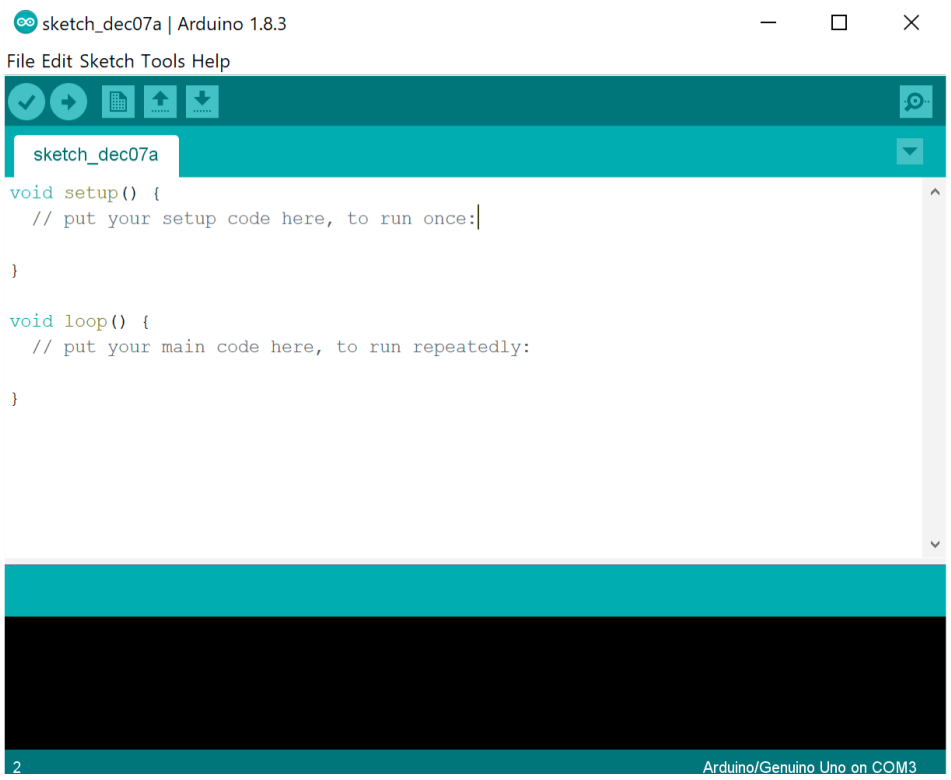


Adaptador de corriente

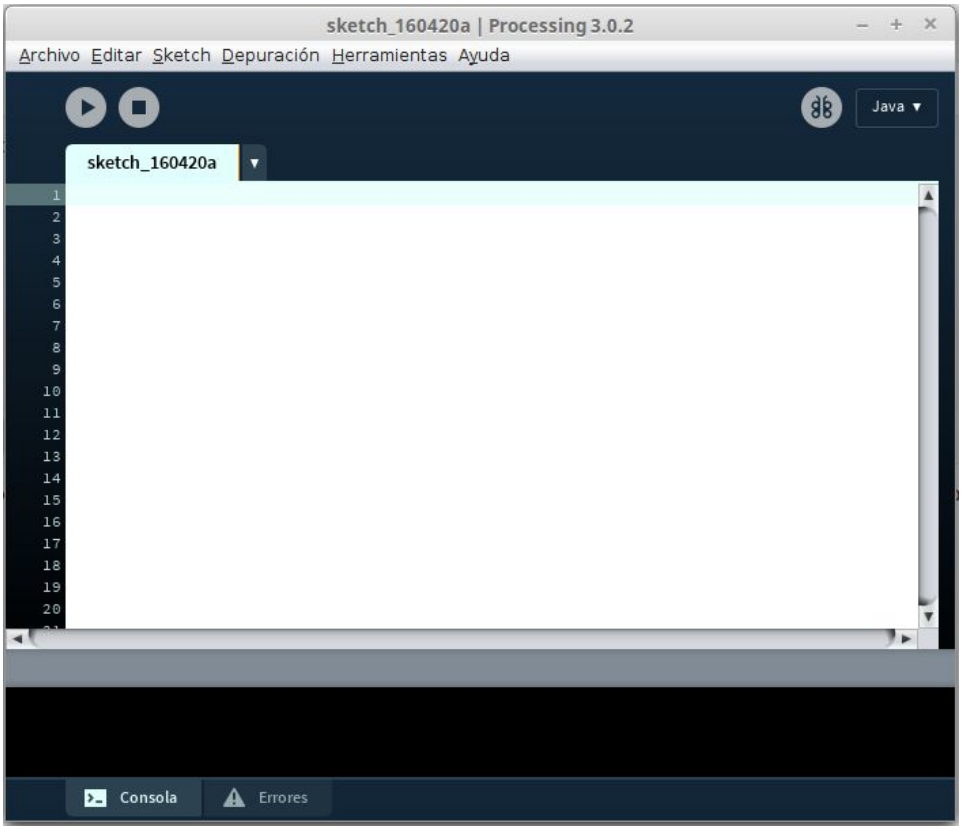
Resistencia de 470 Ω

Tubo Termoretractil

Portátil



IDE de Arduino



IDE de Processing

6. Bibliography

Gantt diagram:
https://drive.google.com/file/d/1O_1kFdXLJOsQPwHXmrrH7M0RmVbp8mlC/view?usp=sharing

https://docs.google.com/spreadsheets/d/1QItiecW_VToMj9kCl3f43bB_uUImqC8hqw0JoBvNnh4/edit?usp=sharing

- ***Arduino Links***
  http://soloelectronicos.com/2015/08/23/fabriquese-su-propio-ambilight/
  http://extra-tecnologia.blogspot.com/2017/05/crear-tu-propio-ambilight-para-pc-con.html
  https://www.askix.com/hacer-tu-propio-ambilight-tv-usando-arduino_2.html#title
  https://geekytheory.com/internet-de-las-cosas-parte-2-subir-los-datos-a-una-base-de-datos
  https://giltesa.com/2015/05/14/guardar-en-un-servidor-web-informacion-enviada-desde-un-arduino
  https://www.youtube.com/watch?v=UhYu0k2woRM
  http://yomaker.com/wp-content/uploads/2017/10/arduino-y-WS2812B-conexion.jpg
  https://howtomechatronics.com/tutorials/arduino/how-to-control-ws2812b-individually-addressable-leds-using-arduino/
  https://forum.arduino.cc/index.php?topic=527501.0
  https://www.youtube.com/watch?v=UhYu0k2woRM
  https://www.youtube.com/watch?v=9hJyyUTflXA
  https://github.com/dmadison/Adalight-FastLED/blob/master/Arduino/LEDstream_FastLED/LEDstream_FastLED.ino
  https://hyperion-project.org/threads/diy-amblight-project-guide-hyperion-ws2801-ws2812b-apa102.8/
  http://www.electronoobs.com/eng_arduino_tut29.php
  https://www.pjrc.com/teensy/td_libs_OctoWS2811.html#addr
  https://github.com/gdsports/p5-js-ws281x
  https://www.tweaking4all.com/hardware/arduino/arduino-ws2812-led/

- ***Links processing***

  https://forum.processing.org/two/discussion/5618/crop-image#Item_26
  https://forum.processing.org/two/discussion/18333/check-color-of-pixel-and-around-it
  https://forum.processing.org/two/discussion/16226/detecting-color
  https://www.youtube.com/watch?v=XN6fqib8AZk
  https://forum.processing.org/one/topic/basic-colour-detection-help.html
  https://funprogramming.org/81-How-to-read-the-color-of-a-pixel.html
  https://processing.org/reference/get_.html
  https://funprogramming.org/81-How-to-read-the-color-of-a-pixel.html
  https://github.com/danteali/Ambilight
  https://forum.processing.org/two/discussion/9556/improve-ambilight-arduino-code
  https://discourse.processing.org/t/a-universal-ambilight/1962/2
  https://oscarliang.com/diy-tv-ambilight-arduino-processing-ozilight-1/
  https://captain-slow.dk/2011/02/26/ambilight-with-arduino-and-processing/
  https://www.instructables.com/id/Ambilight-DFMirage-super-Fast-Screen-Capture-and-P/

- **Links connect**

  https://www.youtube.com/watch?v=7iMMUO0J3h4
  https://forum.arduino.cc/index.php?topic=107319.0
  https://www.youtube.com/watch?v=g0pSfyXOXj8
  https://www.youtube.com/watch?v=yOMgIntmmnA
  https://forum.arduino.cc/index.php?topic=527501.0
  https://forum.arduino.cc/index.php?topic=396450.0
  https://www.youtube.com/watch?v=XosM_kdaha0&list=PLWQQswW6kqpWQh64YIyLdDBcHjdSs16U4&index=3&t=0s
  http://panamahitek.com/arduino-processing-parte-ii-comunicacion-serial/
  https://www.luisllamas.es/guardar-variables-entre-reinicios-con-arduino-y-la-memoria-no-volatil-eeprom/
  http://www.educachip.com/como-usar-la-memoria-eeprom-de-arduino/
  https://leantec.es/blog/35_eeprom-arduino.html
  https://arduino.stackexchange.com/questions/12699/arduino-processing-serial-comunication-losing-data
  https://medium.com/@rjrshr/arduino-based-pc-ambient-lighting-bb370b0b64f1
  https://stackoverflow.com/questions/24853683/arduino-serial-erratic-behaviour-missing-characters-and-occasionally-the-whole-t
  https://forums.ni.com/t5/LabVIEW/losing-data-from-serial-communication-with-arduino/td-p/2626385?profile.language=en
  https://arduino.stackexchange.com/questions/12699/arduino-processing-serial-comunication-losing-data
  https://stackoverflow.com/questions/9072320/split-string-into-string-array