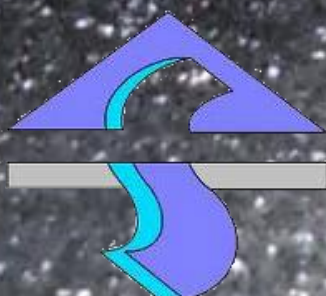


# DISSENY, PROGRAMACIÓ I CONSTRUCCIÓ D'UN PROTOTIP DE SCALEXTRIC INMIND



INS Alcarràs

*Departament de Tecnologia*

Autor: Miquel Ribes Sanjuan

Tutor: Jesús Campa Falcón

2n Batxillerat A

10/02/2016







# ABSTRACT

L'objectiu d'aquest projecte és el disseny, programació i construcció d'un prototip de Scalextric Inmind. Un Scalextric Inmind és un Scalextric controlat amb la ment, amb el nostre nivell de concentració, captat a través d'un casc anomenat NeuroSky Mindwave Headset. Aquesta concentració, programada amb llenguatge Arduino, fa accionar el nostre Scalextric, de forma que, al final d'aquest treball, obtindrem, a part de molta experiència en el camp de l'electrònica i la programació, una sistema diferent i original de gaudir d'un clàssic de l'entreteniment.

El objetivo de este proyecto es el diseño, programación y construcción de un prototipo de Scalextric Inmind. Un Scalextric Inmind es un Scalextric controlado por la mente, con nuestro nivel de concentración, captado a través de un casco llamado NeuroSky Mindwave Headset. Esta concentración, programada en lenguaje Arduino, accionará nuestro Scalextric, de manera que, al final de este trabajo, obtendremos, aparte de mucha experiencia en el campo de la electrónica y la programación, un sistema diferente y original de disfrutar de un clásico del entretenimiento.

The aim of this project is the design, programming and construction of a prototype of Scalextric Inmind. A Scalextric Inmind is a Scalextric controlled by the mind, with our concentration's level, captured through a headset called NeuroSky Mindwave Headset. This concentration, programmed in Arduino language, will activate our Scalextric, so, at the end of this project, we will get, apart from a lot of experience in the field of electronics and programming, a different and original system for enjoying a classical of the entertainment.

# ÍNDEX

<b>1. INTRODUCCIÓ .....</b>	<b>6</b>
1.1 MOTIVACIONS .....	7
1.2 METODOLOGIA .....	7
1.3 OBJECTIUS .....	8
<b>2. DESENVOLUPAMENT .....</b>	<b>10</b>
2.1 ONES CEREBRALS .....	11
2.1.1 QUÈ SÓN? .....	11
2.1.2 TIPUS D'ONES CEREBRALS .....	11
2.1.3 QUINES ONES INTERPRETA EL HEADSET? .....	13
2.2 BCI (BRAIN-COMPUTER INTERFACES) .....	13
2.2.1 QUÈ ÉS UNA BCI? .....	13
2.2.2 HISTÒRIA DE LES BCI .....	14
2.2.3 COM FUNCIONA UNA BCI? .....	15
2.3 ENTORN DE PROGRAMACIÓ: ARDUINO .....	17
2.3.1 QUÈ ÉS? .....	17
2.3.2 ORIGEN .....	18
2.3.3 SOFTWARE/HARDWARE .....	18
2.3.4 PARTS DEL HARDWARE ARDUINO .....	22
2.3.4.1 PLACA .....	22
2.3.4.2 ACCESSORIS .....	23
2.3.5 TIPUS DE TAULETES ARDUINO .....	25
2.3.5.1 OFICIALS .....	25
2.3.5.2 NO OFICIALS .....	26
2.4 SLOT (SCALEXTRIC) .....	26
2.4.1 COM FUNCIONA ELÈCTRICAMENT? .....	27
2.4.1.1 POWERBASE .....	27
2.4.1.2 COMANDAMENT .....	28
2.4.1.3 COTXE .....	30
2.5 DISSENY DEL SCALEXTRIC INMIND .....	32
2.5.1 NEUROSKY MINDWAVE HEADSET → PROCESSING .....	32
2.5.2 PROCESSING → ARDUINO .....	35
2.5.3 ARDUINO → SCALEXTRIC .....	36
<b>3. CONCLUSIONS .....</b>	<b>44</b>
<b>4. WEBGRAFIA .....</b>	<b>48</b>
<b>5. ANNEX .....</b>	<b>54</b>
5.1 CODIS DE PROGRAMACIÓ .....	55
5.1.1 PROCESSING NEUROSKY MASTER .....	56
5.1.2 PROGRAMES AÏLLATS .....	75
5.1.2.1 PROCESSING .....	75
5.1.2.2 ARDUINO .....	76

5.1.3 PROCESSING NEUROSKEY MASTER + PROGRAMA AÏLLAT .....	77
5.1.4 ARDUINO 10 LEDS .....	80
5.1.5 ARDUINO → SCALEXTRIC .....	87
<b>5.2 COMPONENTS DEL SCALEXTRIC INMIND .....</b>	<b>90</b>
5.2.1 NEUROSKEY MINDWAVE HEADSET .....	90
5.2.2 ENTORN DE PROGRAMACIÓ: PROCESSING .....	92
5.2.3 ENTORN DE PROGRAMACIÓ: ARDUINO .....	93
5.2.4 PLACA ELECTRÒNICA ARDUINO → SCALEXTRIC .....	94
5.2.5 SCALEXTRIC .....	95
<b><u>6. AGRAÏMENTS .....</u></b>	<b><u>96</u></b>

# **1. INTRODUCCIÓ**



## 1.1 MOTIVACIONS

Els motius pels quals vaig decidir fer el meu Treball de Recerca sobre aquest tema són diversos.

En primer lloc, des de sempre he estat interessat en l'àmbit de la informàtica i les noves tecnologies. Aquesta afició es va concretar en les classes d'informàtica de 4t d'ESO i, sobretot, 1r de Batxillerat, on ens vam endinsar en el món de la programació. A l'hora d'escollir el tema pel meu TdR, vaig pensar que volia fer que tots els coneixements assolits en el món de la programació, principalment en llenguatge Arduino, tinguessin una aplicació pràctica i tangible, així que vaig escollir realitzar un projecte del camp de la informàtica.

Un cop decidit això, el meu tutor va proposar-me la idea de programar un prototip de Scalextric Inmind, un Scalextric controlat amb la ment. Em va semblar una idea realment engrescadora, perquè complia amb els meus desitjos d'aplicar el que havia après de programació en els últims anys, i d'aprendre'n molt més.

Tot i que, de bon inici, em semblava una idea de projecte bastant complicada, vaig decidir tirar endavant el projecte, gràcies també a l'empenta final que em va suposar parlar amb un estudiant d'Enginyeria Informàtica de la UdL, el Jordi Virgili, que exposava un prototip de Scalextric Inmind a l'institut Guindàvols, en unes jornades on l'Institut d'Alcarràs també formava part. Ell em va donar algun consell, i, sobretot, va dir-me que el projecte era totalment realitzable, així que vaig decidir fer-lo.

## 1.2 METODOLOGIA

La metodologia que he seguit per realitzar aquest treball és, en primer lloc, separar-lo en dos camps: el pràctic i el teòric.

En el camp pràctic, que era bàsicament arribar a construir el Scalextric Inmind, el procés ha seguit un ordre: Primer, vaig buscar informació sobre els materials i aparells a utilitzar, seleccionar els adequats i adquirir-los. Després, amb els materials i aparells ja obtinguts, calia començar a programar-los, tot superant les nombroses complicacions sorgides, com la inexperiència en el món de les BCI o els coneixements insuficients en el camp de la programació.

En el camp teòric, el mètode seguit ha sigut el d'estructurar la teoria en diversos apartats (Ones cerebrals, BCI<sup>1</sup>, Arduino i Scalextric), buscar la informació adequada i redactar-la.

A més, dins de la teoria, a l'apartat 2.5 Disseny del Scalextric Inmind, hi ha una explicació detallada del procediment seguit per desenvolupar la part pràctica, pas a pas.

## 1.3 OBJECTIUS

Finalment, i donant pas ja al que és el treball en si, m'agradaria enumerar alguns dels objectius que buscava assolir amb la realització d'aquest Treball de Recerca:

- Adquirir experiència en el món de la programació en general.
- Realitzar un projecte complex amb el llenguatge de programació Arduino, al qual vaig ser introduït durant les classes d'informàtica de 1r de Batxillerat.
- Introduir-me en el món de les BCI, o interfícies cervell-ordinador, un camp del qual no en tinc ni formació ni experiència.
- Buscar una forma nova i original de gaudir del Slot, col·loquialment anomenat Scalextric, un clàssic de l'entreteniment que tothom coneix i que tothom ha tingut alguna vegada al seu davant.

---

<sup>1</sup> BCI: Brain-Computer Interfaces o interfícies cervell-ordinador.

-Aprendre a cercar informació i seleccionar l'adequada, ja siguin components que comprar pel treball com dades que consultar

-Finalment, i ja com a objectiu personal, realitzar un projecte situat en l'àmbit de la informàtica, camp en el qual m'agradaria seguir estudiant i que, si tot va bé, així serà durant els pròxims anys d'estudis universitaris.

## **2. DESENVOLUPAMENT**

## 2.1 ONES CEREBRALS

### 2.1.1 QUÈ SÓN?

El cervell és el centre del sistema nerviós, que ens permet controlar i coordinar el nostre comportament i les nostres activitats mentals (la memòria, les emocions, l'atenció...). També coordina els cinc sentits (vista, oïda, gust, tacte, olfacte), controla el cicle del son, i una infinitat de funcions més. Ara bé, tot això ja ho sabíem. La qüestió és que al fer totes aquestes funcions, emet unes ones elèctriques de magnituds molt petites, el que s'anomenen ones cerebrals.

El cervell, per tant, manté una activitat bioelèctrica. Aquesta activitat bioelèctrica ve donada per les comunicacions entre les neurones, que s'envien senyals elèctrics per *comunicar-se*. Part d'aquests senyals elèctrics surten fora del cervell, i es poden mesurar mitjançant un electroencefalograma (en endavant, EEG).

Segons la seva freqüència (nombre d'oscil·lacions d'una ona en un segon), els científics distingeixen entre 4 tipus d'ones:


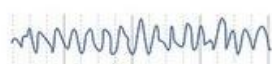


### 2.1.2 TIPUS D'ONES CEREBRALS

A part de diferenciar-se per la seva freqüència, cada tipus d'ona està associat amb una determinada activitat cerebral. De major a menor freqüència, els 4 tipus d'ones són:

- **Ones Beta:** La seva freqüència se situa entre els 14 i els 30-35 Hz. Són les que es produeixen quan el cervell està despert i treballant intensament, ja sigui estudiant, tenint una conversa, resolen problemes, etc.
- **Ones Alpha:** La seva freqüència se situa entre 8 i 14 Hz. Són les que ens indiquen que l'individu està despert però en un estat de relaxació, d'activitat mental baixa. Es produeixen quan una persona està descansant, passejant, etc.

- **Ones Theta:** La seva freqüència se situa entre 4 i 8 Hz. Són les que es produeixen en un estat de calma profunda. Per exemple, es produeixen quan ens quedem “a la lluna”, en un estat en què executem tasques automatitzades per les quals no necessitem tenir un nivell d'atenció i consciència de l'acció. Per exemple, en les repetitives tasques d'un operari en una cadena de muntatge. Es diu que en aquest estat semi inconscient afavoreix la imaginació i la inspiració creativa.

### ONDAS CEREBRALES

<p>Beta (14 - 30 Hz)</p> 	<p>Alfa (8-13.99 Hz)</p> 
<p>Theta (4 - 7.99 Hz)</p> 	<p>Delta (0.1 - 3.99 Hz)</p> 

Tipus d'ones cerebrals <sup>[1]</sup>

- **Ones Delta:** La seva freqüència se situa entre 1,5 i 4Hz. No poden arribar a 0, ja que significaria que el cervell no funciona i, per tant, està mort. Són les que es produeixen en un estat de somni profund.

Es pot considerar que hi ha un cinquè tipus d'ona, les **ones gamma**, la freqüència de les quals se situa entre els 30-50 Hz, però hi ha una gran varietat d'opinions sobre si s'han de considerar un tipus d'ona cerebral. No hi ha unanimitat en la comunitat científica, per tant, no es poden encabir en la classificació anterior.

També és important remarcar que mentre es produeix un tipus d'ona cerebral, la resta es segueixen produint, però en uns valors molt menors en relació al tipus d'ona predominant.

### 2.1.3 QUINES ONES INTERPRETA EL HEADSET?

El NeuroSky Mindwave Headset (o simplement "casc"), que és el dispositiu que he adquirit per dur a terme aquest treball, és capaç de captar el 4 tipus d'ones cerebrals: Delta, Theta, Alfa, Beta, a més de captar les ones Gamma i fer algunes diferenciacions entre ones d'un mateix tipus (per exemple, Low Beta i High Beta, entre d'altres). A més també detecta els parpellejos. Però el que realment ens importa és que capta el nivell d'atenció, que és el que utilitzarem per moure el Scalextric.

## 2.2 BCI (Brain-Computer Interfaces)

### 2.2.1 QUÈ ÉS UNA BCI?

Les interfícies cervell-ordinador (en anglès Brain-Computer Interfaces o **BCI**), també anomenades Mind-Machine Interfaces (MMI), o Brain-Machine Interfaces (BMI), són unes tecnologies que es basen en la captació de les ones cerebrals explicades en l'apartat anterior per tal de ser processades i interpretades per un ordinador. Per tant, fan possible tant l'anàlisi d'aquestes ones cerebrals, com el control d'un ordinador (i el que pugui controlar aquest ordinador) a través del nostre pensament.

Les BCI tenen una gran varietat d'aplicacions, des d'aplicacions mèdiques com poder moure una cadira de rodes, en el cas de persones amb discapacitats motores, o moure una neuropròtesi, és a dir, poder moure un braç o una cama robòtica connectada al nostre sistema nerviós.

A part d'aquestes grans aplicacions, les BCI també s'han fet un lloc en l'àmbit de l'entreteniment, per la possibilitat d'interaccionar amb un ordinador competint amb algú altre utilitzant només la nostra ment. L'exemple més conegut és el del dispositiu Mindball <sup>[2]</sup>. El Mindball és un joc en el qual cada jugador ha d'intentar moure la pilota cap al terreny contrari. La pilota es mou en funció del grau de relaxació dels jugadors. Per tant, guanya el jugador que aconsegueix relaxar-se més.



Aplicació de les BCI: Mindball <sup>[2]</sup>

Aquest és només un exemple de les aplicacions pràctiques de les BCI. Un altre exemple és l'anomenat Scalextric Inmind. Dissenyar i construir un model de Scalextric Inmind, d'una forma menys professional que l'original, és l'objectiu d'aquest treball. Més endavant explicaré què és un Scalextric Inmind.

Ara, toca fer una breu introducció històrica al món de les BCI.

## 2.2.2 HISTÒRIA DE LES BCI

El món de les BCI té els seus inicis l'any 1924 quan el neuròleg alemany Hans Berger va deduir que existien ones elèctriques en el nostre cervell, les ones cerebrals, que es podien captar si s'instal·laven uns elèctrodes sobre la nostra superfície cranial (el cuir cabellut concretament). Ho va provar i li va donar resultats, va ser el primer electroencefalograma (EEG) de la història. Després d'estudiar aquest fenomen durant anys, l'any 1929 el va publicar. Malgrat el gran avenç que suposava en el camp de la medicina, no se li va



donar el premi Nobel de Medicina, motiu que podria haver-lo dut a la depressió i suïcidi l'any 1941.

Tot i que aquest pot ser considerat l'inici de les BCI, no va ser fins a la dècada dels 70 quan es va començar a investigar en aquest àmbit. La investigació anava especialment encarada a finalitats mèdiques, com la implantació de pròtesis neuronals per recuperar l'audició vista o mobilitat perdudes o danyades. Aquestes pròtesis van començar a implantar-se als anys 90.

A partir d'aquest fet, les BCI van començar a créixer en tots els àmbits, i això va comportar que l'any 2001 nasqués la competició bianual BCI, en la que els sistemes BCI del món competeixen entre ells.

Bé, coneixent, a grans trets, la història dels sistemes Brain-Computer Interface, anem a analitzar el funcionament d'aquests sistemes.

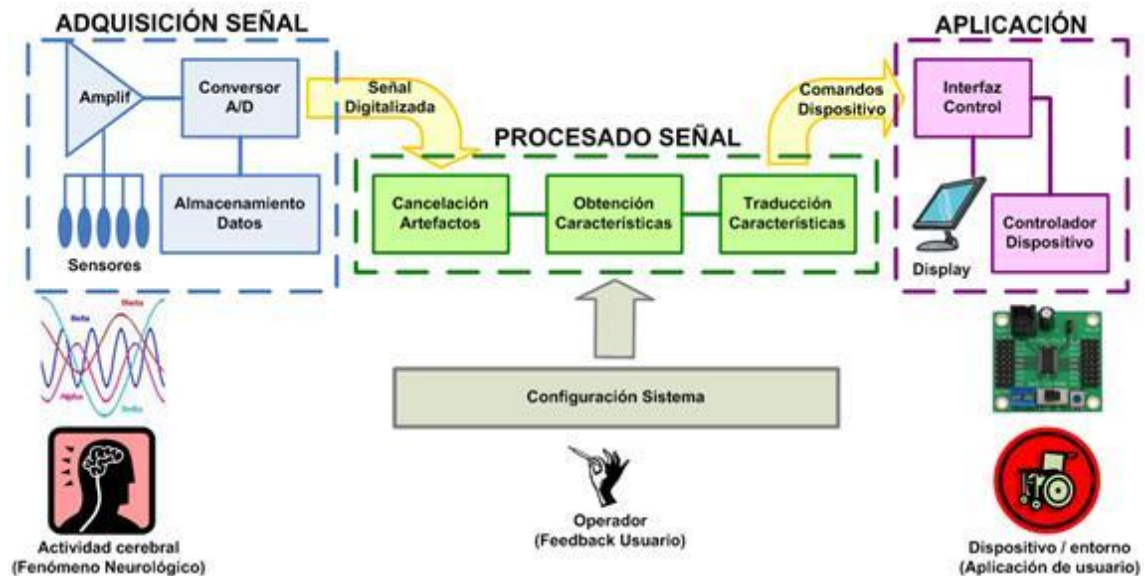
### 2.2.3 COM FUNCIONA UNA BCI?

No totes les BCI funcionen de la mateixa forma. No funciona igual la BCI en la que es basa aquest treball, el NeuroSky Mindwave Headset [3] que he adquirit, que és dels més bàsics del mercat, que els sistemes d'electroencefalografia més avançats.



NeuroSky Mindwave Headset [3]

De totes maneres, totes les BCI segueixen més o menys la següent estructura:



Model de funcionament d'una BCI [4]

En aquest model poder diferenciar-hi 3 grans blocs funcionals:

**-Adquisició del senyal:** Registra l'activitat elèctrica del cervell. Aquesta activitat o senyal, és captada pels sensors (en el nostre cas, el sensor), és amplificada i convertida en un senyal digital per tal que pugui ser processada per l'ordinador. També pot fer la funció d'emmagatzemar dades, però no és indispensable. Aquesta adquisició del senyal es pot fer mitjançant dos tipus de sistemes; els sistemes invasius, en els que la captació es fa directament del cervell i, per tant, es fa per mitjà d'una intervenció quirúrgica; i els sistemes no invasius, en els que l'activitat elèctrica es capta en la superfície cranial.

**-Processament del senyal:** En aquest bloc es rep el senyal en forma digital i es transforma en "ordres" que l'ordinador pot entendre. En aquest bloc s'hi distingeixen 3 etapes.

En primer lloc, la cancel·lació d'artefactes, és a dir, l'eliminació de tots aquells "sorolls" bioelèctrics (per exemple, els moviments oculars o musculars) que poden contaminar el senyal que necessitem. En segon lloc, l'obtenció de característiques, que es basa a traduir el senyal d'entrada en les

característiques que estem buscant (el nivell de relaxació, per exemple). Per últim, la traducció d'aquestes característiques o descodificació, que transforma aquestes característiques en comandes que l'ordinador pot entendre.

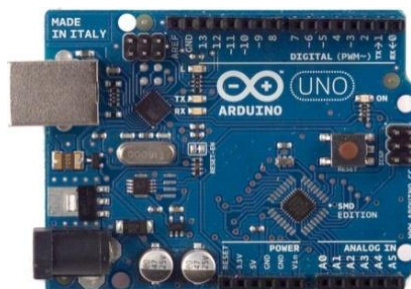
**-Aplicació:** Aquest bloc rep les comandes o “ordres” de l'ordinador i executa les accions que se li han comparat. Per entendre-ho millor, és la pilota en el cas del Mindball o l'activació del cotxe en el Scalextric Inmind.

## 2.3 ENTORN DE PROGRAMACIÓ: ARDUINO

### 2.3.1 QUÈ ÉS?

Segons la seva pàgina web oficial, Arduino “és una plataforma electrònica de codi obert basada en un software i un hardware fàcils d'utilitzar. Està dirigit a qualsevol persona que fa projectes interactius”.

En unes paraules més planeres, Arduino és una forma de programació en què tenim una placa electrònica, com la de la imatge, que té diverses entrades i sortides analògiques i digitals, així com una entrada USB. Aquesta placa és programable, de forma que, connectant la placa a l'ordinador, podem dissenyar projectes en els quals aquestes entrades i sortides, amb els components que s'hi poden connectar (LEDs, bronzidors, sensors de llum, so, distància o temperatura, relés, o fins i tot una placa electrònica que faci de comandament en un Scalextric) puguin interaccionar de la forma que hem programat.



Placa Arduino UNO <sup>[5]</sup>

A més, Arduino també pot interaccionar amb altres programes informàtics com Processing, un entorn de programació amb un llenguatge molt similar que se sol utilitzar per fer representacions gràfiques. De fet, en aquest treball es produeix aquesta interacció entre Arduino i Processing, però d'una forma diferent de la típica: mentre normalment Processing dibuixa una gràfica amb les dades que capten els sensors de la placa Arduino, en aquest treball, és Arduino qui actua accionant el Scalextric a través de les dades que capta Processing.

De totes maneres, l'entorn de programació Arduino és un entorn a l'alça degut principalment a què no és necessària una gran experiència en el món de la programació per utilitzar-lo (tot i que és recomanable tenir un mínim d'experiència) i que, gràcies a ell, es poden realitzar projectes molt variats i de totes les complexitats, des d'engegar i aturar un LED amb un botó a controlar l'accionament d'un Scalextric Inmind.

### 2.3.2 ORIGEN

L'entorn de programació Arduino va néixer l'any 2005 com un projecte de dos estudiants de l' *Interaction Design Institute Ivrea*, a Ivrea, al nord d'Itàlia. La idea consistia en fabricar un dispositiu per controlar projectes de disseny d'interacció en un nivell no excessivament alt, un nivell d'estudiant.

És a dir, crear un dispositiu en el qual poguéssim connectar diversos tipus de components com per exemple sensors, interruptors, LEDs, motors, relés... i en el que poguéssim programar de quina forma interaccionen aquests elements.

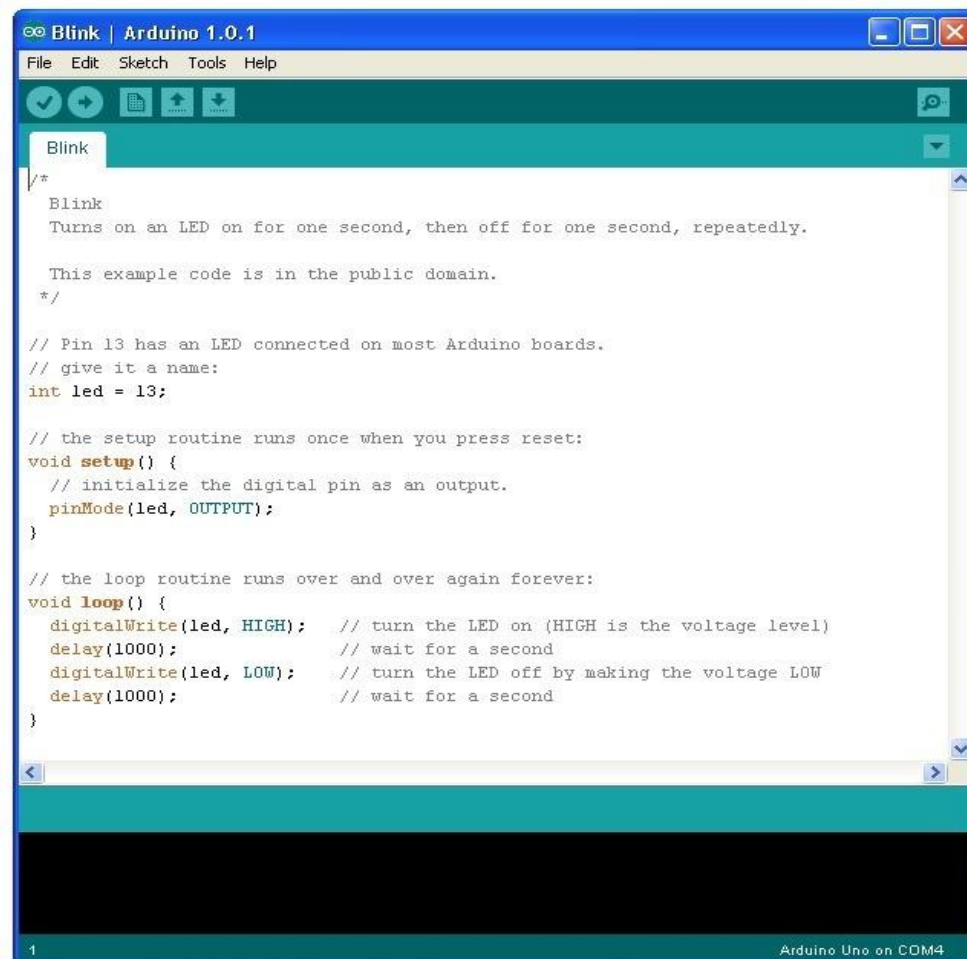
### 2.3.3 SOFTWARE/HARDWARE

Quan parlem d'Arduino, hem de tenir clar que ens podem referir a dos conceptes diferents però relacionats:

El primer, és Arduino com a hardware, és a dir, Arduino com la placa electrònica programable que s'explicarà en les següents pàgines. Aquesta

placa, amb totes les seves aplicacions, no té cap sentit si no està programada. I per programar-la, ho hem de fer amb un entorn de programació, el qual també s'anomena Arduino.

L'entorn de programació Arduino més utilitzat és el que es basa en el llenguatge Wiring, que està inspirat en el llenguatge C.



Entorn de programació Arduino <sup>[6]</sup>

Aquest llenguatge C és dels més importants actualment, i n'hi han derivat altres llenguatges de programació molt utilitzats com C++, Java, PHP o el mateix Wiring. És un entorn molt estandarditzat, tots els llenguatges basats en llenguatge C són molt similars, com es pot veure en les següents imatges.

```
#include<iostream>
using namespace std;
int main()
{
    int palindrome, reverse=0;
    cout<<"Enter number: ";
    cin>>palindrome;

    int num=0, key=palindrome;
    for(int i=1; palindrome!=0; i++){

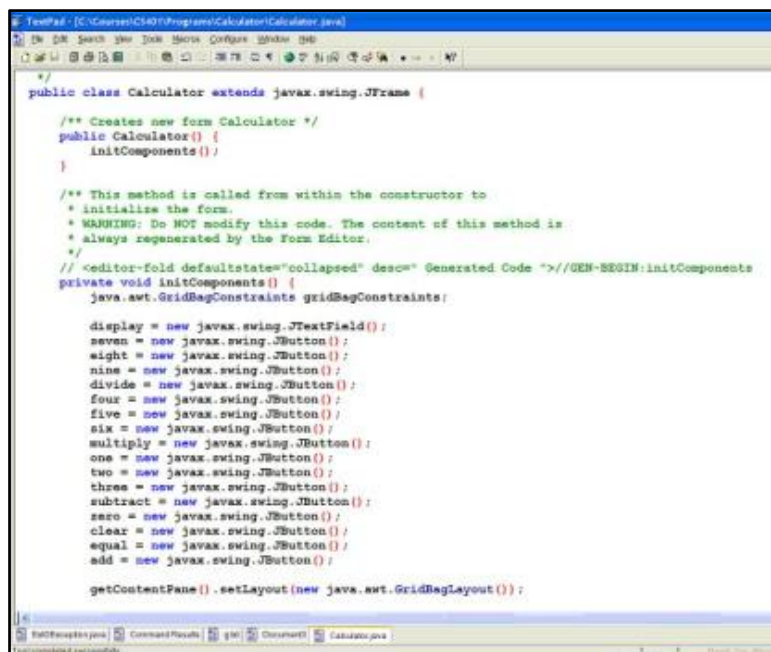
        num=palindrome%10;
        palindrome=palindrome/10;
        reverse=num+(reverse*10);

    }

    if(reverse==key){
        cout<<key<<" is a Palindrome Number";
    }
    else{
        cout<<key<<"is NOT a Palindrome Number";
    }

    return 0;
}
```

Entorn de programació C++ <sup>[7]</sup>



```

public class Calculator extends javax.swing.JFrame {

    /** Creates new form Calculator */
    public Calculator() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN: initComponents
    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        display = new javax.swing.JTextField();
        seven = new javax.swing.JButton();
        eight = new javax.swing.JButton();
        nine = new javax.swing.JButton();
        divide = new javax.swing.JButton();
        four = new javax.swing.JButton();
        five = new javax.swing.JButton();
        six = new javax.swing.JButton();
        multiply = new javax.swing.JButton();
        one = new javax.swing.JButton();
        two = new javax.swing.JButton();
        three = new javax.swing.JButton();
        subtract = new javax.swing.JButton();
        zero = new javax.swing.JButton();
        clear = new javax.swing.JButton();
        equal = new javax.swing.JButton();
        add = new javax.swing.JButton();

        getContentPane().setLayout(new java.awt.GridBagLayout());
    }
}

```

Entorn de programació Java <sup>[8]</sup>

```

192 /
193
194 function PublishFlightTwitter($tid){
195     global $user;
196     global $twitterObj;
197
198     $query = "SELECT depart_time, airport.city AS depart, airport2.city AS arrive FROM travel
199     $query .= "LEFT JOIN airport ON airport.id = travel.depart_airport ";
200     $query .= "LEFT JOIN airport AS airport2 ON airport2.id = travel.arrive_airport ";
201     $query .= "WHERE user=" . $user->id() . " AND travel.id=$tid";
202     $result = mysql_query($query);
203     if ($row = @mysql_fetch_array($result, MYSQL_ASSOC)) {
204         if (isset($_SESSION['oauth_token'])) {
205             $twitterObj->setToken($_SESSION['oauth_token'], $_SESSION['oauth_token_secret']);
206             $twitterInfo = $twitterObj->get('/account/verify_credentials.json');
207         } else {
208             echo "no oauth code";
209         }
210         if (intval($twitterInfo->id_str) == $user->twitterid()) {
211             $text = "Having a flight from " . $row['depart'] . " to " . $row['arrive'] . " @ " . date("d-m-Y");
212             $twitterObj->post_statuses_update(array('status' => $text));
213
214             $query = "UPDATE travel SET tw_posted=1 WHERE user=" . $user->id() . " AND id=$tid";
215             mysql_query($query);
216         }
217     }
218 }
219
220 function PublishMeetingTwitter($mid){
221     global $user;
222     global $twitterObj;
223
224     $query = "SELECT meeting.time, meeting.title, airport.displayname FROM participant ";
225     $query .= "LEFT JOIN meeting ON meeting.id = participant.meeting ";
226     $query .= "LEFT JOIN airport ON airport.id = meeting.airport ";
227     $query .= "WHERE participant.user=" . $user->id() . " AND meeting.id=$mid";

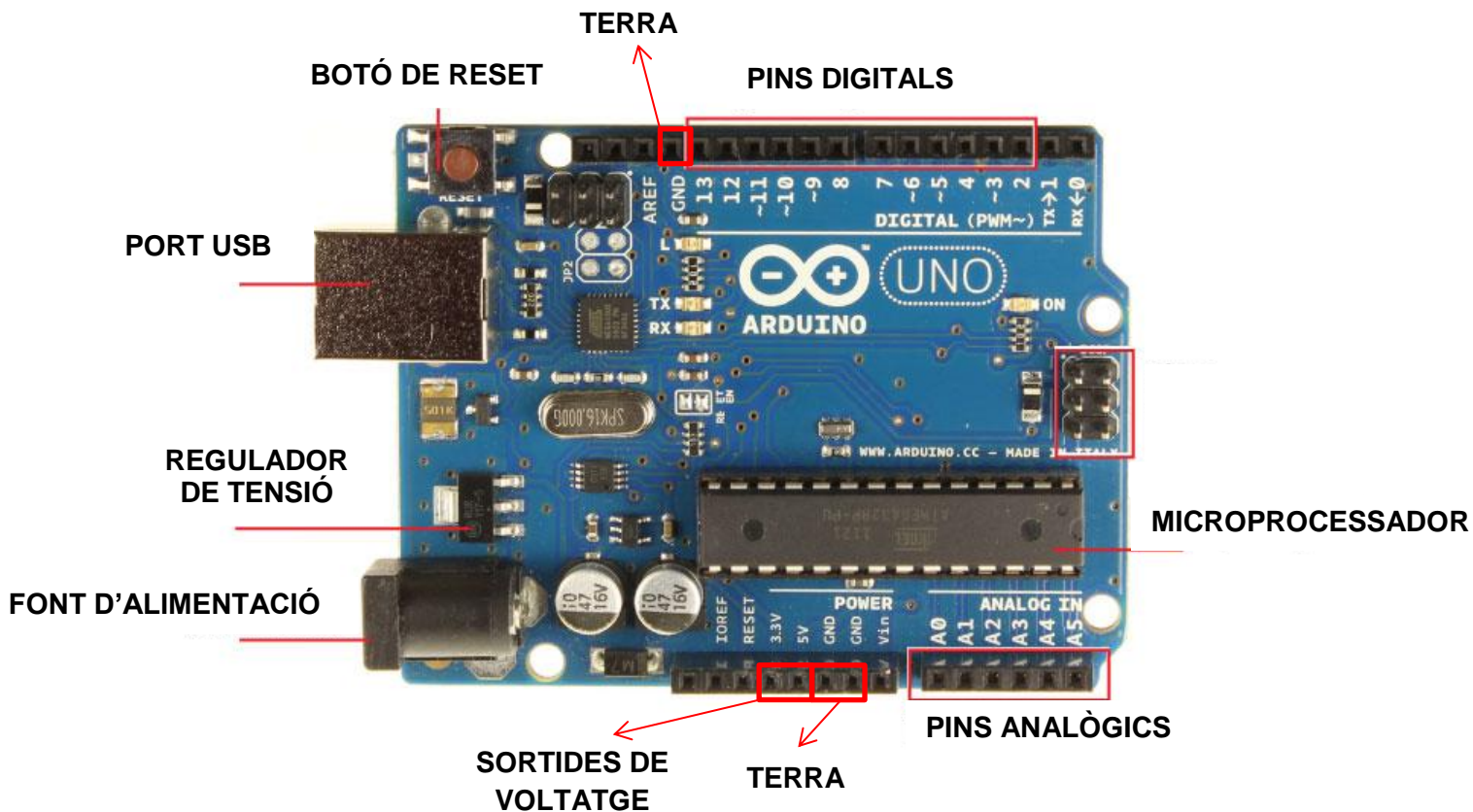
```

Entorn de programació PHP <sup>[9]</sup>



## 2.3.4 PARTS DEL HARDWARE ARDUINO

### 2.3.4.1 PLACA



**-PORT USB:** Connexió per on, a través d'un cable USB, es connecta la placa Arduino a la placa. Si la placa està connectada a l'ordinador, també exerceix com a font d'alimentació.

**-REGULADOR DE TENSIÓ:** Dispositiu que manté constant el valor el voltatge que té la placa.

**-FONT D'ALIMENTACIÓ:** Quan la placa no està connectada a un ordinador, entrada per on se li subministra el voltatge a la placa.

**-MICROPROCESSADOR:** Executa el programa que conté la placa, aquell que hem programat anteriorment.



**-PINS ANALÒGICS:** Entrades (o sortides) per les quals la placa emet (o llegeix) un voltatge. Aquest voltatge pot ser de qualsevol valor entremig de 0 i 5 V, ambdós inclosos. Hi ha 6 pins analògics (A0-A5).

**-PINS DIGITALS:** Entrades (o sortides) per les quals la placa emet (o llegeix) un voltatge. Aquest voltatge pot ser de 0 V (LOW) o de 5 V (HIGH), sense valors entremitjos. Hi ha 14 pins digitals (0-13).

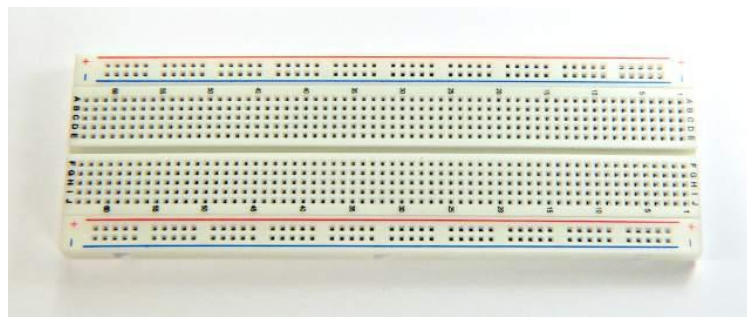
**-SORTIDES DE VOLTATGE:** Ofereixen un voltatge constant de 3,3 V o de 5V (normalment s'utilitza el de 5V).

**-TERRA:** Ens ofereix un voltatge de 0V, una presa a terra.

### 2.3.4.2 ACCESSORIS

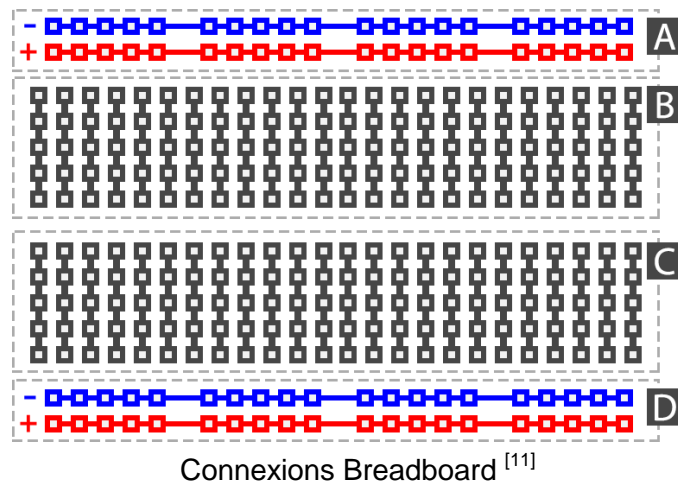
A part de la placa, es necessiten altres components per treballar amb Arduino. Els més importants és el següent:

**-BREADBOARD:** També anomenada *protoboard* o *tauler de pa*. És un tauler d'orificis connectats de la manera que posteriorment explicaré, que serveix per fer les connexions entre els elements del projecte de manera temporal.



Breadboard del Kit Arduino <sup>[10]</sup>

Els orificis estan connectats de la següent manera:



D'aquesta forma, si volem connectar, per exemple, la pota negativa d'un LED amb una de les dues potes d'una resistència, els col·locarem els dos en una de les fileres de 5 orificis negres, i els dos components faran contacte

Normalment, un dels orificis marcats en vermell (+) està connectat amb la sortida de voltatge de 5V de la placa, de manera que tots els orificis vermells tenen un voltatge de 5V.

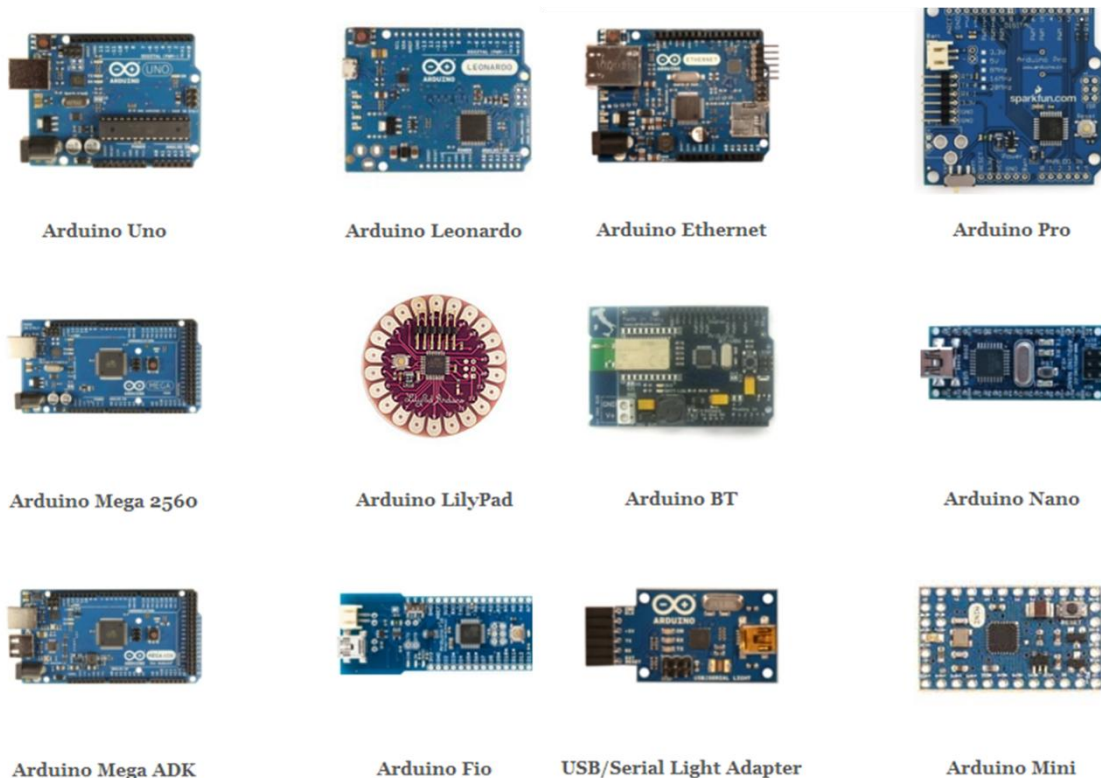
De la mateixa manera, és freqüent connectar un dels orificis marcats en blau (-) amb el GND o "terra" de la placa, de manera que tots els orificis blaus tenen un voltatge de 0V.

D'altra banda, també trobem altres components que són vitals per al correcte funcionament dels projectes que realitzem amb Arduino. Aquests són els **sensors** (de llum, temperatura, distància...), els **actuadors** (LEDs, bronzidors, displays...) o el **cablejat**, entre altres components.

## 2.3.5 TIPUS DE TAULETES ARDUINO

### 2.3.5.1 OFICIALS

Hi ha una gran quantitat de tipus de tauletes Arduino al mercat. En primer lloc, dins de les tauletes oficials es poden distingir entre les *Arduino*, venudes als EEUU, i les *Genuino*, destinades a fora del EEUU. Algunes, tot i que no totes (ni de bon tros) de les plaques Arduino disponibles al mercat són les següents.



Tipus de tauletes Arduino oficials <sup>[12]</sup>

La més comuna és l'*Arduino Uno*, però n'hi ha altres amb aplicacions especials. Així, l'*Arduino Mega 2560* té més pins digitals i analògics, l'*Arduino Ethernet*, que té entrada de cable de xarxa o l'*Arduino Nano*, que té una mida molt inferior.

### 2.3.5.2 NO OFICIALS

En segon lloc trobem les plaques Arduino no oficials. N'hi ha centenars, i són fàcils de reconèixer com a plaques no oficials compatibles a Arduino. Gairebé sempre, els seus noms acaben amb el sufix **-uino**. En són exemples les següents plaques: **AVR.duino U+**, **Brasuino**, **ChibiDuino2**, **Diavolino**, **Freeduino**, **Seeeduino**, **Faraduino**, i un llarg etcètera.

La placa que jo utilitzo en aquest treball també és una placa no oficial però amb les mateixes característiques que una d'oficial. Es tracta d'una **Funduino UNO**, una placa molt semblant a l'*Arduino UNO* i de la qual l'institut en disposa unes quantes unitats.

## 2.4 SLOT (Scalextric)

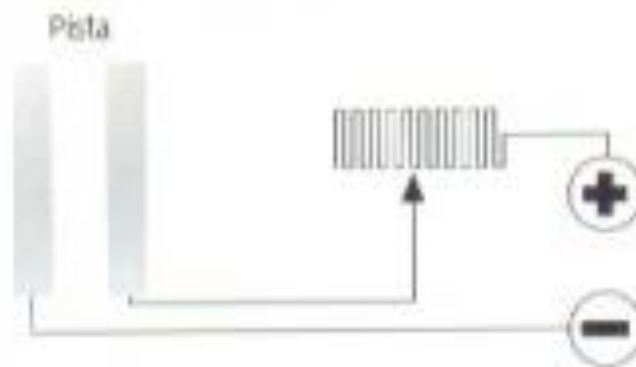
Abans de tot, vull comentar que al llarg d'aquest treball m'he referit al que generalment coneixem com Scalextric tant amb el nom de Slot com el de Scalextric. M'agradaria remarcar que el nom correcte d'aquest esport és Slot, però comunament en diem Scalextric perquè és el nom d'una de les primeres marques de fabricants de slots. Durant els últims 20 anys Scalextric no ha tingut gaire força, però el món no especialitzat ja hem anomenat Scalextric al joc de pistes i cotxes en miniatura, per tant, en aquest treball s'han utilitzat i s'utilitzaran indistintament les dues nomenclatures.

Fet aquest aclariment, també vull dir que el món del Slot és tan extens que donaria per fer un treball de recerca exclusiu per a ell. Per això, dins d'aquest àmbit jo em centraré en el funcionament elèctric d'aquest esport, que és el que realment ens interessa per poder substituir el comandament de Scalextric convencional pel nivell d'atenció del nostre cervell.

## 2.4.1 COM FUNCIONA ELÈCTRICAMENT?

El funcionament elèctric d'un Slot és relativament senzill:

El pol positiu de la font d'alimentació està connectat a un born de la pista, mentre que l'altre born està connectat a "terra", el pol negatiu, 0 V. Entre el pol positiu de la font d'alimentació i el born positiu de la pista s'hi troba el comandament, el funcionament del qual s'explicarà més extensament en endavant. El cotxe aprofita aquesta diferència de potencial entre els dos borns de la pista com a alimentació pel seu motor, cosa que més endavant també s'explicarà.



Esquema elèctric bàsic d'un Scalextric <sup>[13]</sup>

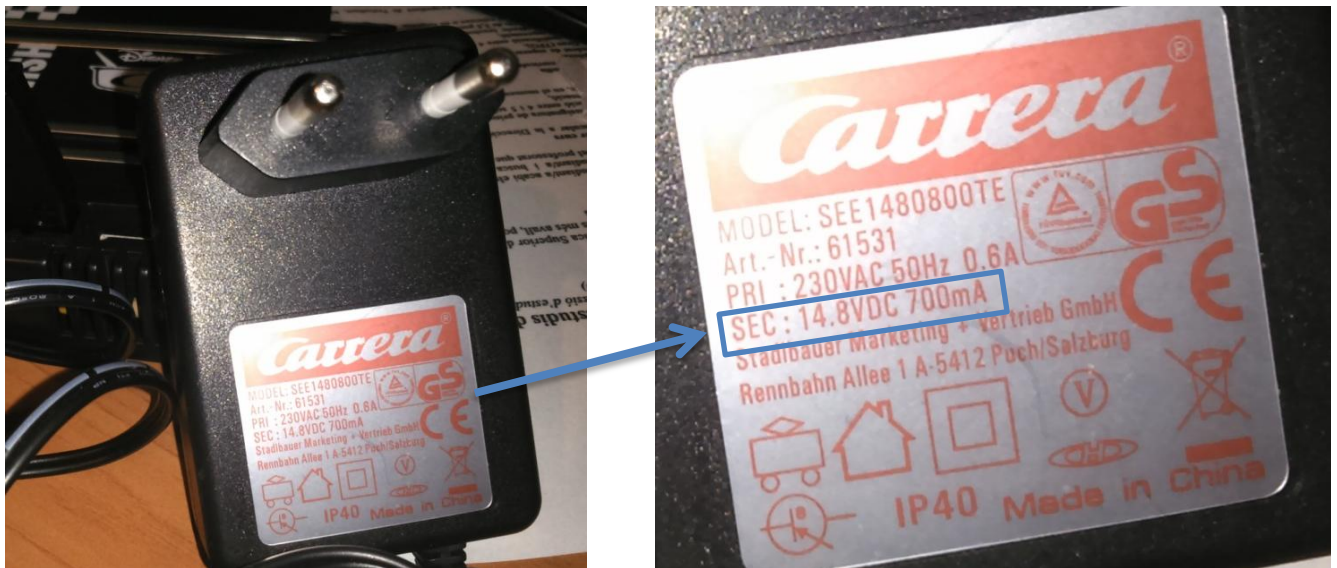
Feta aquesta breu introducció, expliquem el funcionament més detalladament.

### 2.4.1.1 POWERBASE

La *powerbase*, o font d'alimentació, és l'encarregada de donar corrent al circuit elèctric que, al cap i a la fi, és un circuit de Slot.

En primer lloc, és important recordar que els circuits de Slot no funcionen amb els 230 V de corrent altern que arriben als endolls de les nostres cases. De fet, cap electrodomèstic ho fa, tots necessiten un transformador. Aquest transformador té dues grans funcions: transformar el corrent altern en corrent continu i convertir el voltatge i la intensitat rebuts en les adequades per l'electrodomèstic.

Doncs bé, en aquest cas, el transformador del Slot transforma el corrent altern a un corrent continu amb un voltatge de 14.8 V i una intensitat de 700 mA.



Transformador d'un Slot [imatge pròpia]

Fet aquest pas, la powerbase es comporta com una pila de petaca, té un pol positiu i un de negatiu. Així, el pol positiu es connecta amb el rail positiu de la pista, havent passat pel comandament, i el pol negatiu es connecta amb el rail negatiu, tancant així el circuit en el moment en que els dos rails es comuniquin (quan posem el cotxe de pista).

#### 2.4.1.2 COMANDAMENT

El funcionament del comandament del Slot ha variat al llarg dels anys. Al principi era només com un interruptor, que donava o no corrent al circuit. Al llarg dels anys, aquest comandament ha anat evolucionant a un comandament analògic amb el qual ja es podia regular la velocitat, i els últims models ja són comandaments electrònics, que tenen un funcionament bastant diferent dels analògics.

Dit això, jo em centraré en el comandament més habitual, tradicional, el comandament analògic.

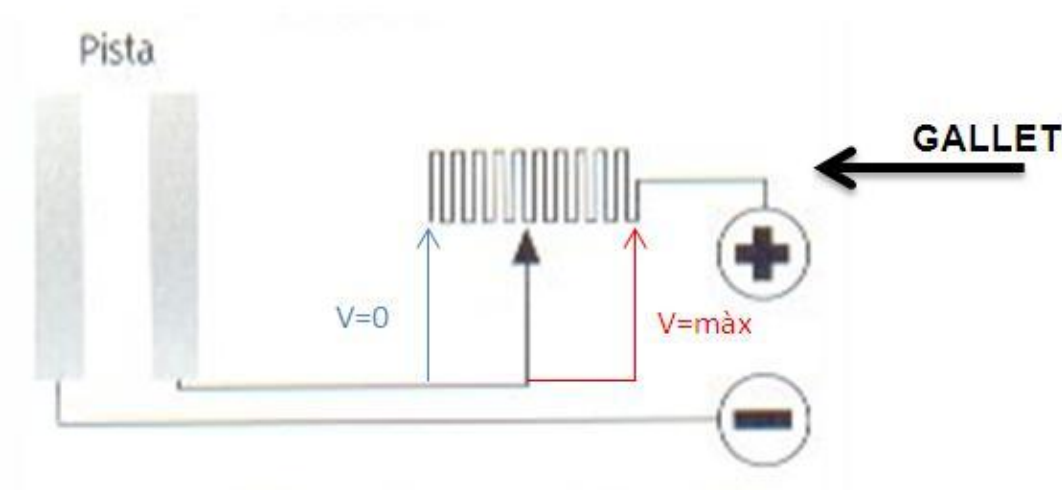


Comandament analògic <sup>[14]</sup>

El funcionament d'aquest comandament és relativament senzill. A l'interior del comandament hi trobem una resistència, que provoca una oposició al pas del corrent elèctric. Un extrem d'aquesta resistència està connectat a la font d'alimentació (al pol positiu), mentre que l'altre rail de la pista està connectat en un punt intermedi de la resistència. Aquest punt variarà en funció de quan premem el gallet del comandament.

Finalment, el pol negatiu de la font d'alimentació està connectat al rail negatiu de la pista.

Amb un esquema s'entén millor:



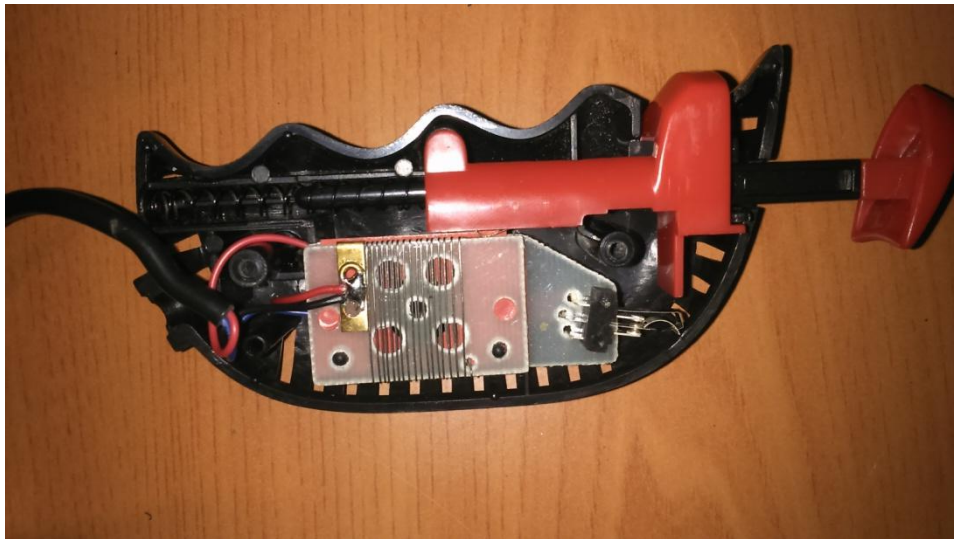
D'aquesta forma, si no premem el gallet (fletxa blava), el corrent elèctric ha de *travessar* tota la resistència i no arriba voltatge a la pista: el cotxe no es mou.



Contràriament, si premem al màxim el gallet (fletxa vermella), el corrent no ha de *travessar* cap fragment de resistència, i tot el voltatge de la font d'alimentació arriba al cotxe, que anirà a la seva velocitat màxima.

Per tant, si premem el gallet en un valor entre el mínim i el màxim, la velocitat del cotxe serà directament proporcional a la distància que hem premut el gallet i inversament proporcional a la quantitat de resistència *travessada*.

Per fer-se una idea més gràfica i realista del funcionament d'aquests tipus de comandament, els més utilitzats actualment, podem veure l'interior d'un d'aquests comandaments, en aquest cas un comandament propi:



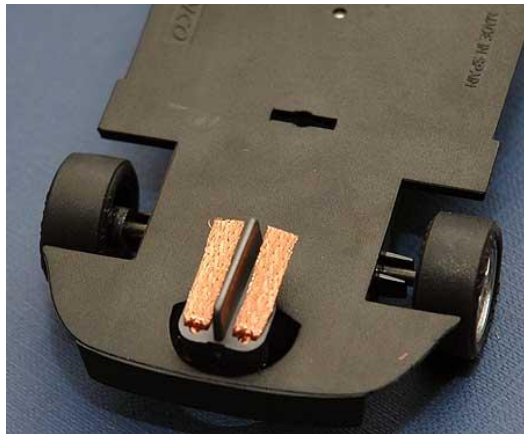
Comandament analògic [imatge pròpia]

### 2.4.1.3 COTXE

El funcionament del cotxe de Slot és també bastant senzill. Com s'ha explicat abans, el cotxe aprofita la diferència de potencial dels dos borns de la pista per fer moure el motor. Així, el cotxe es connecta als borns de la pista mitjançant unes *escombretes* <sup>[15]</sup> d'un material conductor, que pot ser de coure, estany o plata. Segons el material de les escombretes, la conductivitat, la duració i la rigidesa varien.



Per tant, a través de les *escombretes* el rail positiu de la pista es connecta al pol positiu del motor, i el mateix succeeix amb el negatiu, tal com es veu en la imatge<sup>[16]</sup>. Un cop alimentat, el motor transmetrà el moviment a l'eix de les rodes de darrere, que proporcionaran el moviment al cotxe depenent del voltatge que arriba al motor, regulat pel comandament.



*Escombretes del cotxe* <sup>[15]</sup>



*Motor del cotxe* <sup>[16]</sup>

El cotxe té altres parts molt importants, tals com el xassís, la carrosseria, els eixos o l'imant, que és un element optatiu que fa que el cotxe s'agafi més a la pista, tot i que li treu atractiu a les carreres.

## 2.5 DISSENY DEL SCALEXTRIC INMIND

En aquest subapartat explicaré tot el que recorregut que fan les dades que ens proporciona el casc (NeuroSky Mindwave Headset), fruit de la nostra activitat bioelèctrica cerebral, des de que són captades fins que accionen el Scalextric.

Els codis dels programes utilitzats en el prototip de Scalextric Inmind es troben a l'annex d'aquest treball, en el subapartat 5.1 Codis de programació, i s'aniran citant convenientment dins de l'explicació que ve a continuació.

A més, tots els components utilitzats en la realització del prototip de Scalextric Inmind, sobre els quals es parlarà en el pròxim apartat, estan definits i catalogats en l'annex d'aquest treball, en el subapartat 5.2 Components del Scalextric Inmind.

### 2.5.1 NEUROSKY MINDWAVE HEADSET → PROCESSING

Primer de tot, hem d'instal·lar tot el programari que ve amb el “pack” del casc.

Per fer-ho, només s'ha de seguir les instruccions que donava el *driver* o instal·lador del CD que anava inclòs amb el casc. En cas de no tenir el CD, també es pot aconseguir el driver seguint el següent link de descàrrega: <http://dom.cat/r2k>.

Un cop seguides les instruccions, obtindrem principalment dos programes: El *NeuroSky App Central* i el *ThinkGear Connector* (TGC). El primer programa és un executable que ens ofereix diverses aplicacions a realitzar amb el casc, com exercicis de concentració, de meditació o jocs mitjançant els quals podem millorar el nostre nivell d'atenció o meditació.



NeuroSky App Central [imatge pròpia]

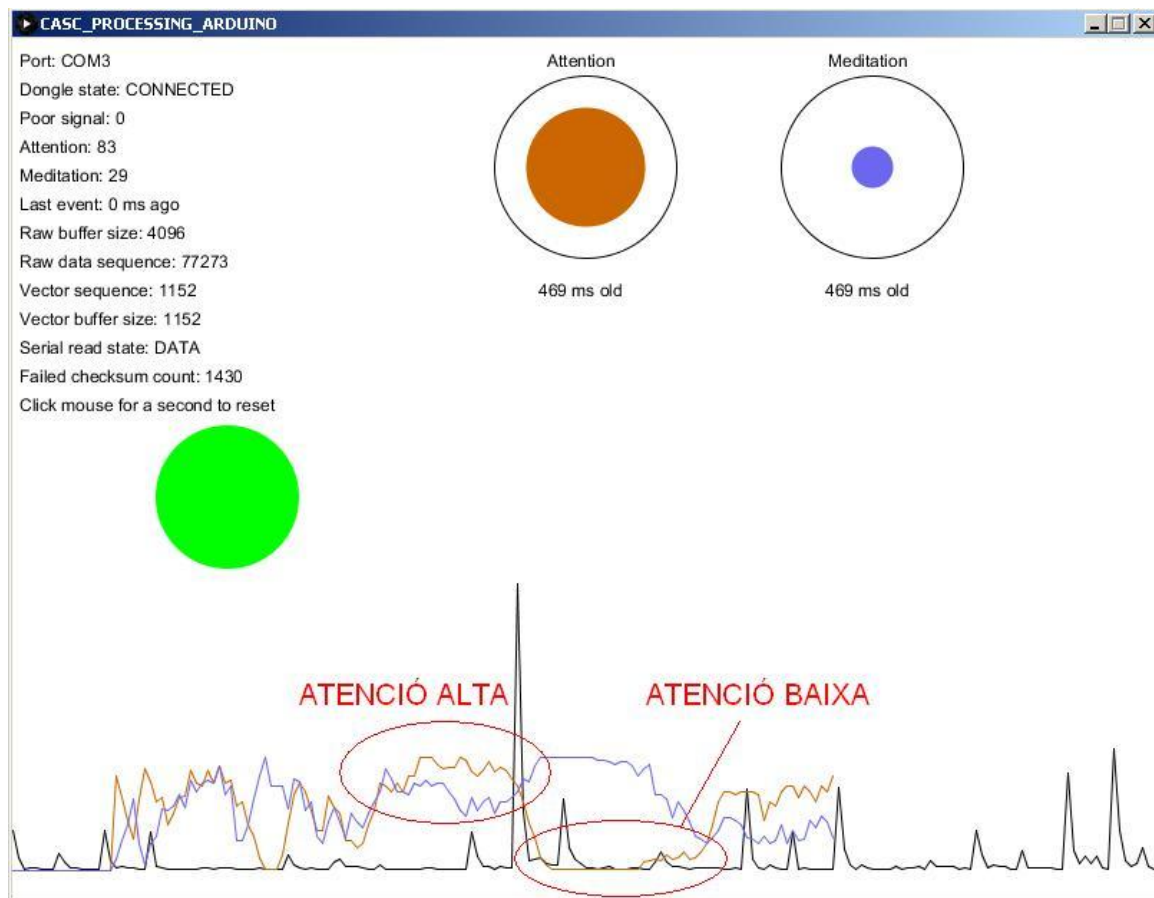
El segon, el que s'anomena *ThinkGear Connector* (TGC), és un programa que s'executa en segon pla mentre utilitzem el casc i que s'encarrega de dirigir les dades del casc des del port sèrie on hem connectat l'USB fins al programari que ens dona per defecte NeuroSky, el *NeuroSky App Central*. La transmissió de dades entre el casc i l'USB es produeix via Bluetooth.

En el casc del Scalextric Inmind, el que fem és interceptar les dades del TGC i tractar-les amb l'entorn de programació Processing. Això es fa de la manera següent:

En primer lloc, NeuroSky no ofereix un programa oficial per poder tractar les dades que rep el casc directament. Bé, sí que l'ofereix, s'anomena *Research Tools* i té un cost de \$500, un preu totalment fora de pressupost.

Així doncs, s'ha hagut de cercar a Internet un programa alternatiu que permetés extreure les dades del casc per tractar-les en un entorn de programació. I després de molt buscar, el programa adient s'anomena *Processing NeuroSky Master*, que és un executable en entorn Processing

mitjançant el qual no només es podia observar els nivells d'atenció i meditació a temps real, sinó que també es podien tractar aquestes dades.



Processing NeuroSky Master [imatge pròpia]

El codi de programació d'aquest programa es pot trobar a l'annex d'aquest treball, en el subapartat 5.1.1 Processing NeuroSky Master.

Òbviament, i com sol passar en el camp de la programació, el programa no va funcionar la primera vegada, donava diversos errors, que buscant informació i, sobretot, gràcies a l'ajuda del meu tutor, van poder ser solucionats.

Fet aquest pas, les dades de la nostra activitat elèctrica cerebral que capta el casc ja poden ser, no només visualitzades a temps real, sinó que també programades per a realitzar l'objectiu final, moure el Scalextric. Per fer-ho, el següent pas és passar les dades de l'entorn Processing a l'entorn Arduino.

## 2.5.2 PROCESSING→ARDUINO

Un cop obtinguda la variable *atenció* (que és un número del 0 al 100), hem d'enviar-la des de *Processing* a l'entorn de programació *Arduino*, que, com ja s'ha explicat en aquest treball, ens permet gestionar components electrònics a través d'un programa informàtic. L'entorn *Processing* ens permet visualitzar les dades de forma gràfica, i els dos programes poden funcionar de forma simultània, de manera que podem veure a la pantalla la nostra atenció i alhora que aquesta funcioni com a *gallet del nostre Scalextric*.

Per tant, hem d'enviar les dades des de Processing a Arduino, de la manera següent:

En primer lloc, és important dir que el que és habitual és transferir dades des d'Arduino a Processing, i no a l'inrevés com necessitem en aquest treball. Per tant, hi ha poca informació a Internet per connectar aquests dos programes en el sentit que necessitem. Per fer-ho, s'ha seguit dos passos ben diferenciats.

El primer pas ha sigut buscar dos executables aïllats, independents, que permetessin enviar dades a Arduino des de Processing. Aquests programes, en els quals Processing genera una variable aleatòria i l'envia a través del port sèrie a Arduino, es poden trobar a l'annex d'aquest treball, en el subapartat 5.1.2 Programes aïllats.

Un cop aquests dos programes connectats funcionaven correctament, el següent pas era incorporar-los al programa amb el qual obteníem la dada *atenció*, el *Processing NeuroSky Master*.

Per fer-ho, no només s'havia d'incloure el programa aïllat de Processing al del casc, sinó que també s'havien d'obrir els ports sèrie adients, fet que va suposar també bastants problemes, però que també van poder ser resolts.

L'executable Processing que captava l'*atenció* juntament amb el programa aïllat que enviava les dades a l'entorn Arduino es pot trobar a l'annex d'aquest treball, en el subapartat 5.1.3 Processing NeuroSky Master + Programa aïllat,

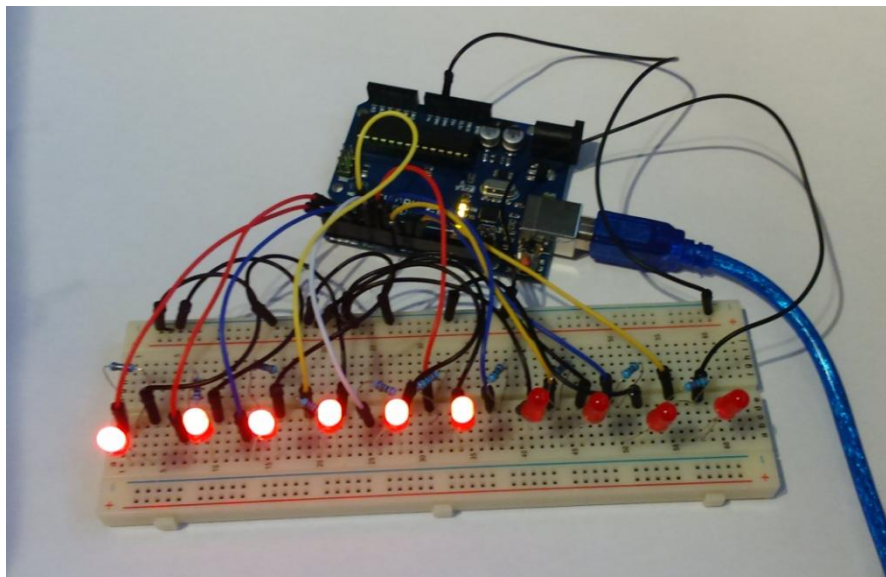
on es remarquen les instruccions del programa aïllat annexionades al programa que capta l'*atenció*.

Un cop la fusió dels dos executables de Processing i el d'Arduino funcionaven correctament i de forma simultània, l'objectiu era, des de l'entorn Arduino, comandar el Scalextric, a través de la dada *atenció* que, a temps real, el *NeuroSky Mindwave Headset* enviava, després de seguir un llarg camí, a la placa Arduino.

### 2.5.3 ARDUINO→SCALEXTRIC

Obtinguda la variable *atenció*, el primer que he fet abans de pensar a fer accionar el Scalextric amb aquesta variable és poder veure, d'alguna forma, aquesta dada representada en l'entorn Arduino.

Per fer-ho, he utilitzat un dels components més habituals en el món de l'electrònica, els LEDs.



10 LEDs, amb les seves resistències, connectats a la placa Arduino [*imatge pròpia*]

Aquests LEDs s'encenien més o menys en funció del nostre nivell d'atenció, a través d'un programa d'Arduino. Sent més concrets, i considerant el rang de la nostra atenció entre 0 i 100; si aquesta atenció era inferior a 10 s'encén 1 LED, si és entre 10 i 19, 2 LEDs, entre 20 i 29, 3 LEDs... Així fins a arribar a una atenció d'entre 90 i 100, que fa engegar els 10 LEDs.

El codi que permetia encendre els LEDs en funció de la nostra atenció es troba a l'annex d'aquest treball, en el subapartat 5.1.4 Arduino 10 LEDs.

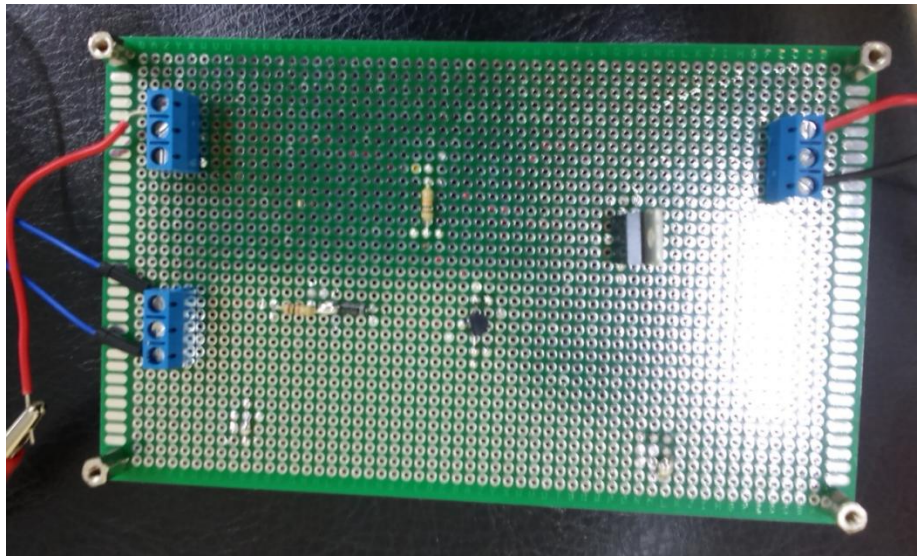
Un cop ja podíem veure representada la dada *atenció* gràcies als 10 LEDs, el següent pas era utilitzar aquesta dada per comandar l'Scalextric. Ara bé, aquest pas tenia certes dificultats: Arduino no ofereix una sortida analògica real (que pugui aportar valors variables de voltatge diferents de 0V i 5V), sinó que ofereix uns pins anomenats PWM.

El PWM, o modulació per amplada de polsos, d'una font d'energia, que és com actua en aquestes circumstàncies la placa Arduino, és una tècnica mitjançant la qual es poden oferir valors "analògics" de voltatge que realment són un conjunt de valors digitals. Així, Arduino, oferint només unes sortides reals de 0V o 5V, pot arribar a oferir un senyal d'1, 2, 3, 4, 4,56 o 2,23 V, per exemple.

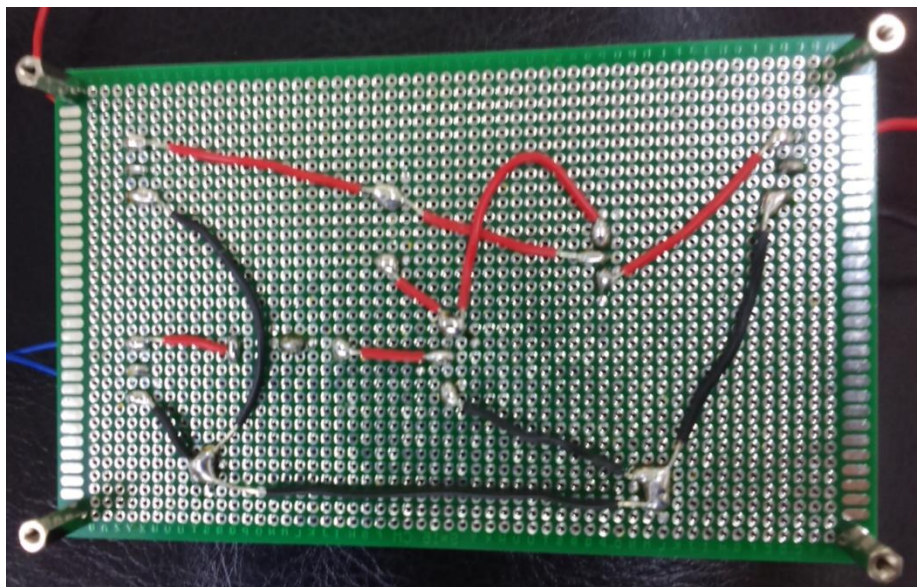
Però és clar, aquesta sortida no podia ser connectada directament al Scalextric, ja que aquest ha de rebre un valor de voltatge *vertader*, no PWM, i, sobretot, perquè el voltatge màxim que pot oferir Arduino és 5V, molt inferior als 7-8V a partir dels quals el cotxe es pot començar a moure.

I és aquí quan apareix una part del Scalextric Inmind que encara no s'havia comentat abans, la **placa electrònica**. Aquesta placa, de manufactura pròpia, és la culpable que puguem controlar el diferencial de voltatge que arriba als rails del Slot a través de la placa Arduino, i també la culpable de l'ajornament d'aquest treball, ja que ha sigut una de les parts que més problemes ha donat.





Placa electrònica Arduino→Scalextric *[imatge pròpia]*



Placa electrònica Arduino→Scalextric (revers) *[imatge pròpia]*

El disseny de la placa és relativament senzill per algú mínimament format en el camp de l'electrònica, però necessita una explicació per als qui teníem molt poques nocions sobre aquest camp. Per entendre el funcionament d'aquesta placa, primer en veurem el seu diagrama:



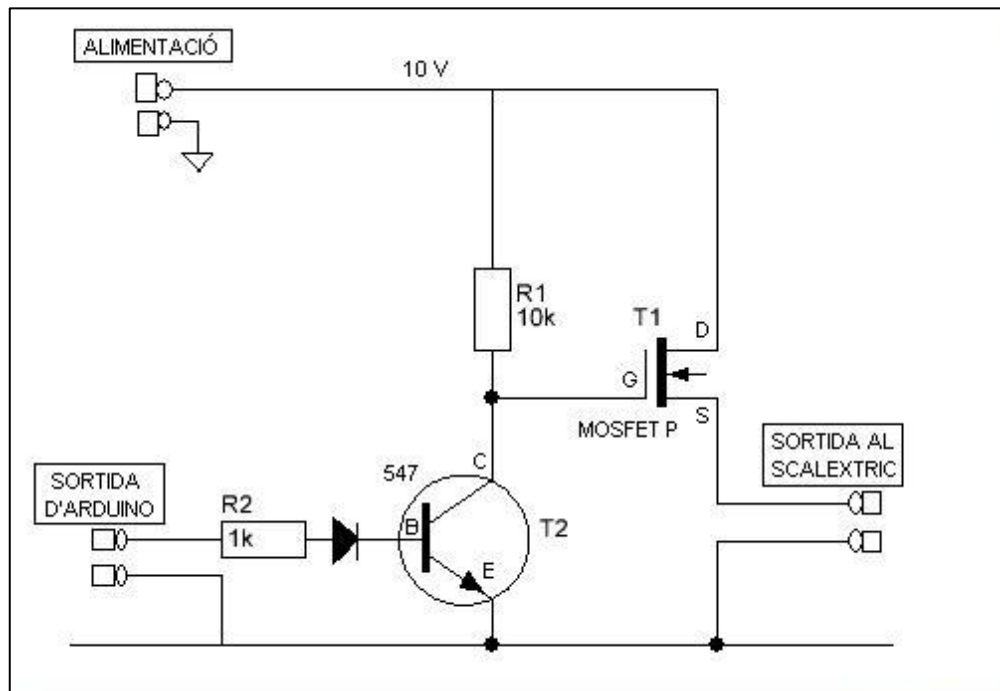


Diagrama de la placa electrònica definitiva [imatge pròpia]

Sota el requadre on indica *ALIMENTACIÓ*, hi trobem un connector al qual es connecten els 10 V que ens ofereix la font d'alimentació. El pol negatiu en connecta amb tots els altres negatius: el de la placa Arduino, el de la font d'alimentació i el del Scalextric estan units.

Continuant amb l'explicació, arriben 10 V al transistor T1, que és un MOSFET P. Aquesta diferència de potencial recorrerà el transistor (arribant a la sortida al Scalextric) en funció del voltatge que arriba a la porta G del MOSFET, que actua com una aixeta reguladora del voltatge. Aquesta *aixeta* es regula en funció de, ara sí, el voltatge que oferim des d'Arduino, que abans passa per una resistència, un díode i un altre transistor (547).

El voltatge que acabi sortint de l'*aixeta* és el que anirà a parar al Scalextric. Aquesta sortida es connectarà com si es tractés d'un comandament de Slot, connectant així el pol positiu de la sortida de la placa amb el positiu del comandament del Slot, i connectant el negatiu de la placa amb el negatiu del Slot.

L'anterior és una explicació bastant escassa i poc tècnica, però és suficient per entendre la finalitat de la placa electrònica.

També s'ha de recordar que la idea de disseny de la placa no és meu, sinó que pertany al Jordi Virgili, el noi nomenat en l'apartat de motivacions del començament. Ell em va fer arribar un diagrama molt semblant a aquest, però més complex. La placa inicial que seguia aquell diagrama no va funcionar, així que vam decidir simplificar-la, tot i que s'ha d'agraciar la col·laboració i consell del Jordi en aquest apartat.

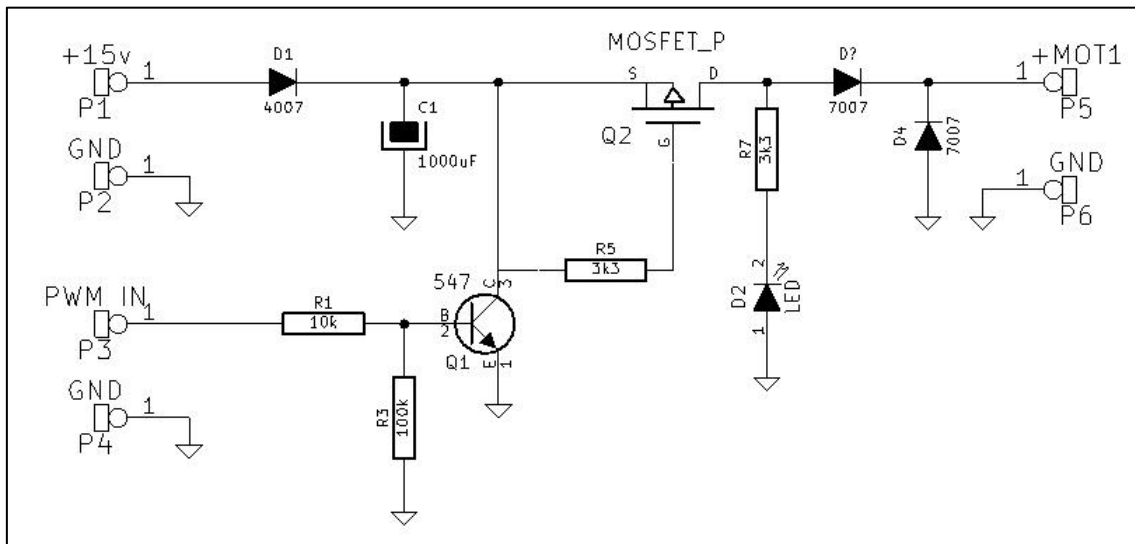


Diagrama inicial, que finalment no va ser utilitzat <sup>[17]</sup>

Així doncs, el que falta és programar la placa Arduino perquè ofereixi una sortida analògica (en PWM, recordem) que, a través de la placa electrònica, reguli el voltatge que arriba als rails de la pista. Aquest programa estarà inclòs dins del programa que abans hem anomenat dels *10 LEDs*, ja que la mateixa tauleta Arduino que encendrà els LEDs en funció de l'*atenció* serà la que porti el voltatge regulat (en PWM) a la placa electrònica.

El codi, amb la seva corresponent explicació, es troba a l'annex d'aquest treball, en el subapartat 5.1.5 Arduino→Scalextric.

Per finalitzar l'explicació del funcionament del prototip de Scalextric Inmind, cal comentar alguns detalls tècnics d'aquest mateix prototip, com la connexió al Scalextric o l'alimentació que rep. Són aspectes menys transcendents, però igualment importants i necessaris:

En el moment de connectar la placa electrònica amb la *powerbase* de Scalextric, hem de tenir en compte que el que s'ha fet en aquest prototip és una connexió de tipus temporal. D'aquesta manera no ha calgut "fer malbé" el Scalextric, i aquest pot seguir funcionant amb el comandament convencional només desconnectant els cables que es veuran en la següent imatge.

La connexió, doncs, és aquesta:

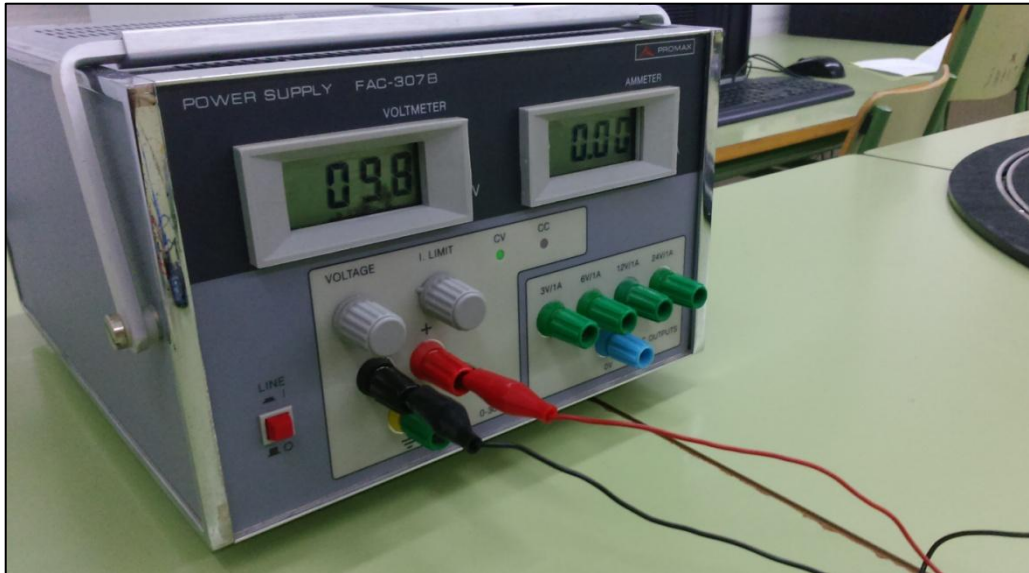


Connexions a la *powerbase* del Scalextric [imatge pròpia]

El cable groc és la sortida positiva de la placa electrònica, i el negre és el *terra*, el neutre, 0 V. A més, a causa del funcionament intern de la *powerbase*, s'ha hagut de curtcircuitar els dos pols destinats al comandament convencional, unint-los amb un cable que també està connectat a *terra*, als 0 volts.

Un altre aspecte a comentar dins del funcionament del prototip és que finalment l'alimentació de la placa electrònica (que ja hem dit que és com el substitut del comandament convencional) no és aportada pel transformador convencional del Scalextric. Ve proporcionada per una font d'alimentació ajustable amb la

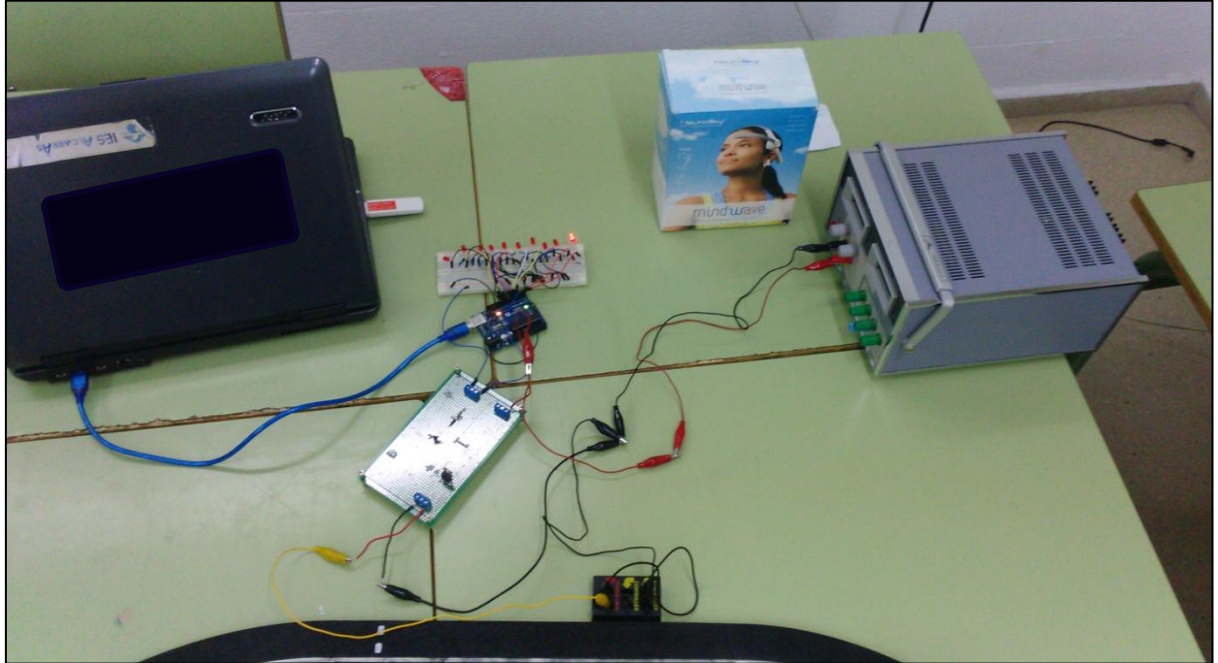
qual es pot regular la diferència de potencial oferta, establir una intensitat màxima per precaució, i que ofereix un major control del prototip.



Font d'alimentació ajustable [imatge pròpia]

Gràcies a aquesta font d'alimentació ajustable, podem establir el voltatge d'alimentació en un valor d'entre 10 i 11 volts, fet que permet al cotxe no sortir-se del rail en cas d'anar a la velocitat màxima. Aquest voltatge pot ser regulat depenent del cotxe que fiquem al Scalextric, ja que no tots tenen la mateixa velocitat pel mateix valor de tensió.

Superat aquest últim pas, el nostre prototip de Scalextric Inmind ja està llest per funcionar.



Prototip complet de Scalextric Inmind [imatge pròpia]

## **3.CONCLUSIONS**

Un cop finalitzat el disseny, programació i construcció del prototip de Scalextric Inmind, puc afirmar que els objectius que em vaig plantejar abans de la realització del treball han estat complerts. He acumulat molta experiència i el prototip finalment ha estat un èxit. Tot i la satisfacció amb el resultat final, també nombraré algunes limitacions i possibles millores del projecte:

Una de les limitacions que té aquest prototip és que només pot funcionar en sistemes operatius que treballin a 32 bits, tals com Windows XP o Windows Vista. En sistemes operatius a 64 bits, com Windows 7 o posteriors, el programa que capta l'atenció no funciona correctament i, en conseqüència, el prototip no funciona. Aquest seria un aspecte a millorar si en algun moment hi hagués un perfeccionament del projecte.

Una altra millora possible és connectar un sensor de distància (disponible en el kit Arduino de l'institut) per poder cronometrar les voltes i poder així fer competicions o reptar-se a un mateix. Aquesta millora, a part d'augmentar la competitivitat entre les persones que provin el prototip, seria un gran sistema per entrenar la nostra capacitat d'atenció.

Dit això, procedeix a exposar les conclusions a les quals he arribat després d'aquest projecte:

En primer lloc, m'he adonat de les innumerables aplicacions que té el camp de la informàtica i la programació. Ja tenia una idea que són uns àmbits amb moltes possibilitats i que cada cop seran més importants en la nostra societat, però el fet de poder tastar-ho en primera persona m'ha fet encoratjar encara més amb l'itinerari d'estudis que tenia planejat seguir.

Comentant més concretament el camp de la programació, vull citar un fenomen que m'ha succeït realitzant el Scalextric Inmind. En la part de connectar el Mindwave Headset amb un entorn de programació, que ha resultat ser finalment Processing, hi ha hagut moltes dificultats. Com a dificultats em refereixo al fet que pots estar 2 o 3 tardes treballant en una idea que al final no acaba servint de res, o estar setmanes sense fer cap avenç tècnic.

Ara bé, també és veritat que totes aquestes complicacions poden arribar a desaparèixer amb una sola tarda. El que vull dir amb això és que el camp de la programació és estrany, ja que no sempre el treball es converteix en recompensa, de la mateixa manera que pots trobar molta recompensa amb poc treball. L'única solució a aquest fenomen és la dedicació i l'ànim de creure que, més aviat o més tard, la recompensa arribarà.

En segon lloc, també he tingut l'oportunitat d'adquirir molts coneixements en el camp de les BCI i de l'electrònica, parts indispensables d'aquest treball. Sent més concret, la construcció de la placa electrònica que connectava Arduino amb el Scalextric m'ha aportat molta informació que de ben segur em serà molt vàlida en un futur.

Aquests coneixements han estat adquirits tant de forma teòrica com pràctica. En aquest sentit, m'agradaria comentar la relació que tenen aquests dos conceptes. Encara que aquest es tracti d'un projecte principal pràctic, tota la investigació teòrica feta abans és vital per no endinsar-se en el treball amb els ulls clucs. A més, tot el subapartat 2.5 Disseny del Scalextric Inmind és clau per arribar a entendre el funcionament d'aquest prototip, fet que demostra que la teoria i la pràctica estan estretament relacionades.

La realització d'aquest treball de recerca també m'ha servit per demostrar-me que, movent-se una mica, hi ha molta gent que té un gran esperit de col·laboració i està disposada a ajudar-te. Ja esmentaré a qui toqui en els agraïments, però la col·laboració desinteressada de gent com el Jordi Virgili, fòrums d'Internet com el d'Arduino, Processing o el d'*Automodelismo Slot*, els amics i el meu tutor han estat claus pel correcte desenvolupament d'aquest treball.



Però, sense restar importància a les conclusions anteriors, la conclusió més important que he extret de la realització d'aquest prototip de Scalextric Inmind és una altra. Sense intencions de ficar-me filosòfic ni res semblant, la conclusió a què he arribat és que els reptes, amb ganes i dedicació, se superen. Per mi era un repte realitzar aquest treball, com de ben segur ho ha estat per tots els meus companys amb els seus respectius treballs. En la consecució d'aquest repte, com en qualsevol altre, les ganes solen estar garantides, tothom vol que les coses li surtin bé. Ara bé, això no passa si no hi ha dedicació. La dedicació, la constància, i fins i tot el sacrifici en alguns aspectes, són vitals per aconseguir els nostres objectius.

## **4. WEBGRAFIA**

<https://ca.wikipedia.org/wiki/Cervell>



<http://neurosky.com/biosensors/eeg-sensor/biosensors/>



<http://www.ub.edu/pa1/node/130>



<http://www.dom.cat/>



<http://www.lacofa.es/blog/2008/12/15/introduccion-a-los-sistemas-brain-computer-interface/>



<http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/>



<http://automodelismoslot.mforos.com/>



<http://forum.arduino.cc/>



<https://forum.processing.org/two/>



[https://es.wikipedia.org/wiki/Modulaci%C3%B3n\\_por\\_ancho\\_de\\_pulsos](https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos)



<http://www.qrcode.es/es/generador-qr-code/>



## IMATGES:

- [1] <http://www.ub.edu/pa1/sites/default/files/Fig.%200.1%20Cuatro%20tipo%20de%20ondas%20cerebrales.jpg>
- [2] <http://www.xplorator.se/store/Mindball.JPG>
- [3] [http://a.tgcdn.net/images/products/additional/large/e9e5\\_neurosky\\_mindwave\\_parts.jpg](http://a.tgcdn.net/images/products/additional/large/e9e5_neurosky_mindwave_parts.jpg)
- [4] <http://www.lacofa.es/blog/2008/12/15/introduccion-a-los-sistemas-brain-computer-interface/>
- [5] <http://i.blogs.es/90d0d7/arduinounosmd450px/original.jpg>
- [6] [https://learn.adafruit.com/system/assets/assets/000/002/147/medium800/learn\\_arduino\\_ide\\_blink.jpg?1396779953](https://learn.adafruit.com/system/assets/assets/000/002/147/medium800/learn_arduino_ide_blink.jpg?1396779953)
- [7] [http://people.cs.pitt.edu/~mehmud/cs401/image/lab5\\_img20.JPG](http://people.cs.pitt.edu/~mehmud/cs401/image/lab5_img20.JPG)
- [8] [http://1.bp.blogspot.com/\\_eBUaURerUs/UjTgc42JfFI/AAAAAAAAACY/RijUFcQs0QM/s1600/palindrome-number-in-c++.GIF](http://1.bp.blogspot.com/_eBUaURerUs/UjTgc42JfFI/AAAAAAAAACY/RijUFcQs0QM/s1600/palindrome-number-in-c++.GIF)
- [9] [http://www.caribitech.com/images/php\\_programming.png](http://www.caribitech.com/images/php_programming.png)
- [10] <https://www.pjrc.com/store/breadboard.jpg>
- [11] [http://www.tweaking4all.com/wpcontent/uploads/2013/12/basic\\_breadboard\\_layout.png](http://www.tweaking4all.com/wpcontent/uploads/2013/12/basic_breadboard_layout.png)
- [12] [http://2.bp.blogspot.com/KPpJ6kznJzI/VCWqmqR\\_D6I/AAAAAAAAABV4/1qag9\\_iihgw/s1600/arduinios.png](http://2.bp.blogspot.com/KPpJ6kznJzI/VCWqmqR_D6I/AAAAAAAAABV4/1qag9_iihgw/s1600/arduinios.png)
- [13] <http://blogslot.aloyshop.com/wpcontent/uploads/2014/07/MandoPulsador.jpg>
- [14] [http://www.slot4ever.com/server/Portal\\_0003451/img/products/mando-standard-scalextric-analogico\\_362938.jpg](http://www.slot4ever.com/server/Portal_0003451/img/products/mando-standard-scalextric-analogico_362938.jpg)
- [15] <http://s5.postimg.org/dtnkrdvl3/0505trencillas08jl1.jpg>

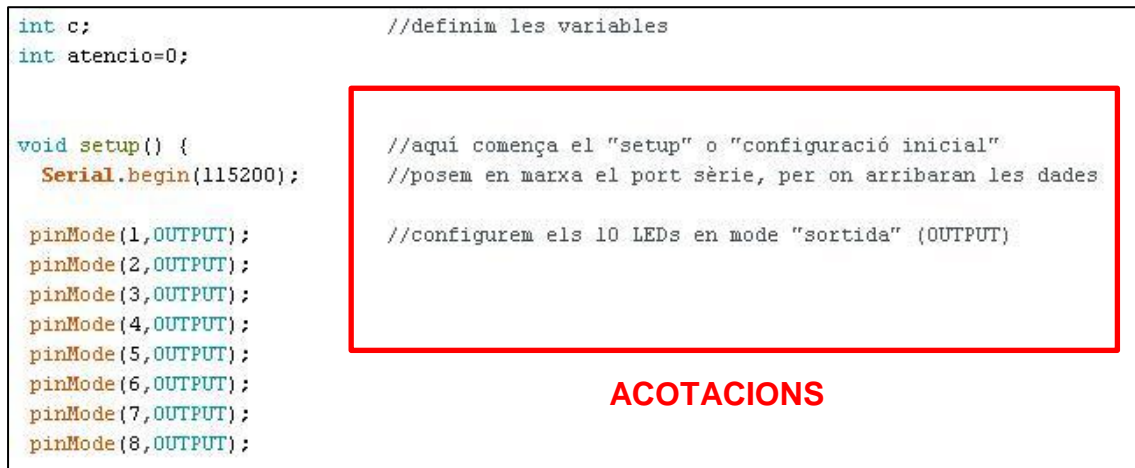
- [16] <http://cloud1.todocoleccion.net/scalextricexin/tc/2015/06/12/08/49810305.jpg>
- [17] <http://virgili.org/wp-content/uploads/2015/03/control-pwm-sch.jpg>

## **5. ANNEX**



## 5.1 CODIS DE PROGRAMACIÓ

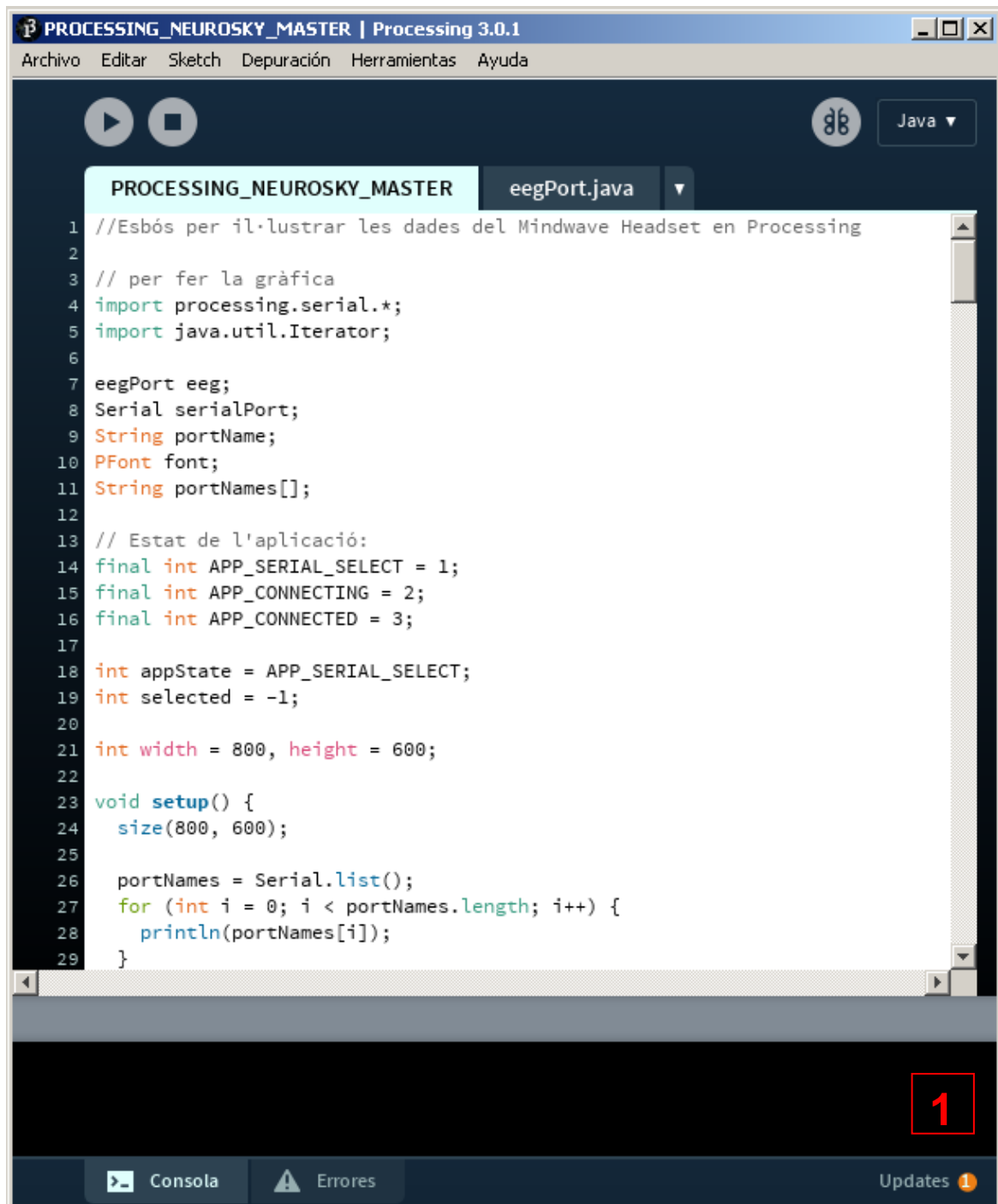
En aquest apartat hi estan els codis de programació de tots els programes utilitzats per la realització del projecte. En cada codi hi ha unes **acotacions** com les de la següent imatge:

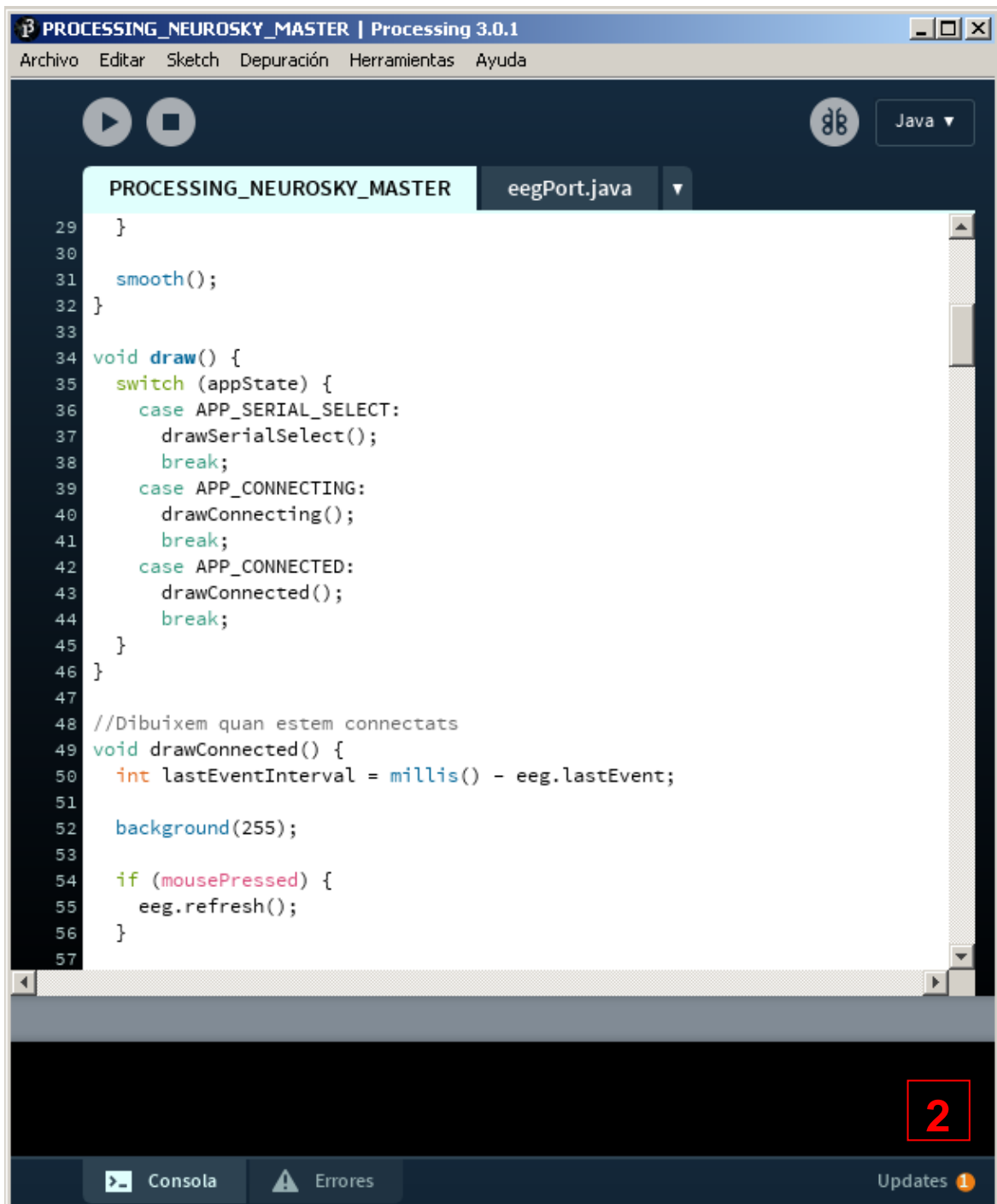


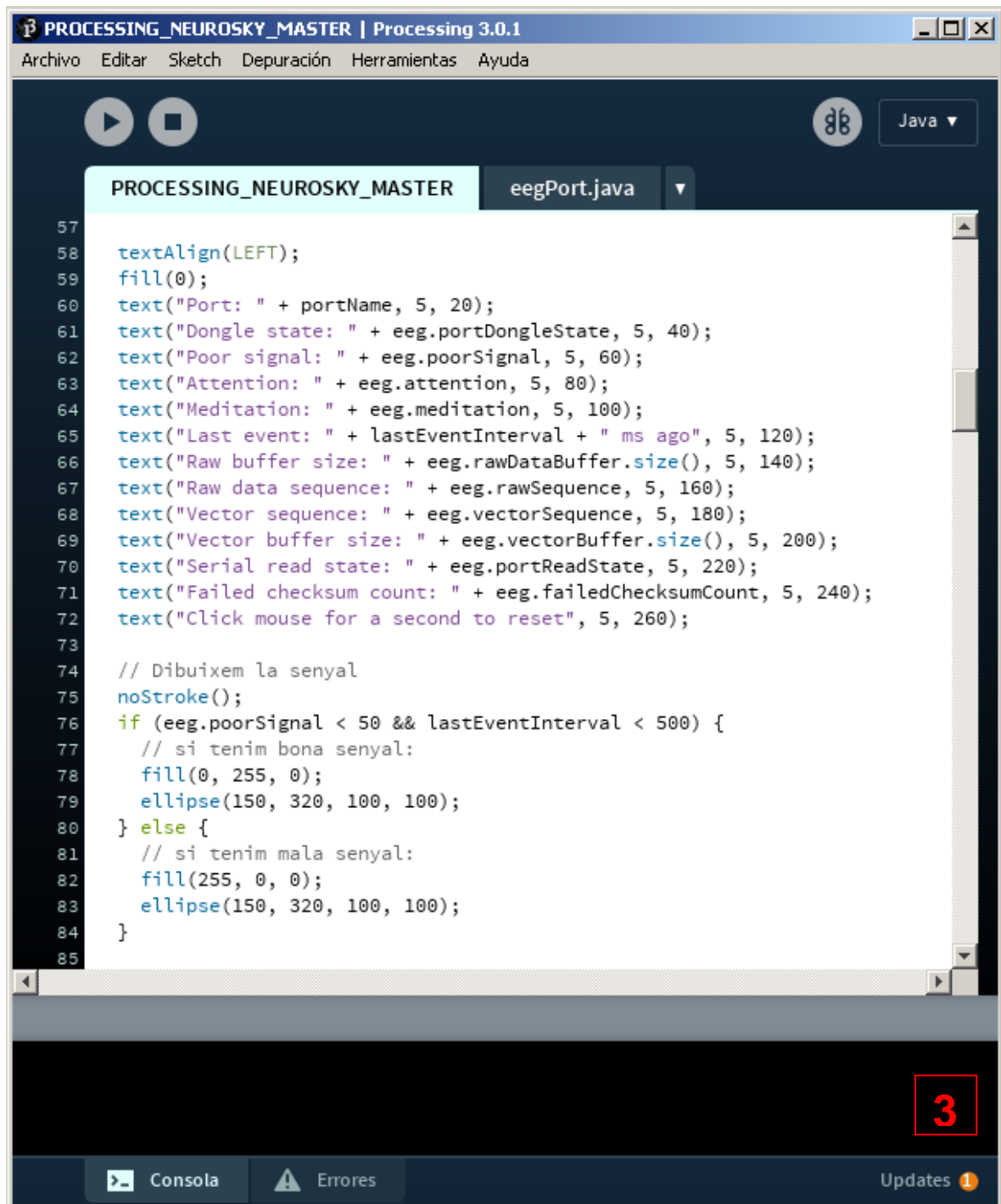
Aquestes acotacions, sempre en color gris i precedides per dues barres (//), són explicacions del mateix codi, comentaris per poder entendre millor el programa, ja que a simple vista és difícil de comprendre. Algunes d'aquestes acotacions aniran acompanyades d'una petita explicació a la mateixa memòria, per tal de fer els aclariments necessaris.

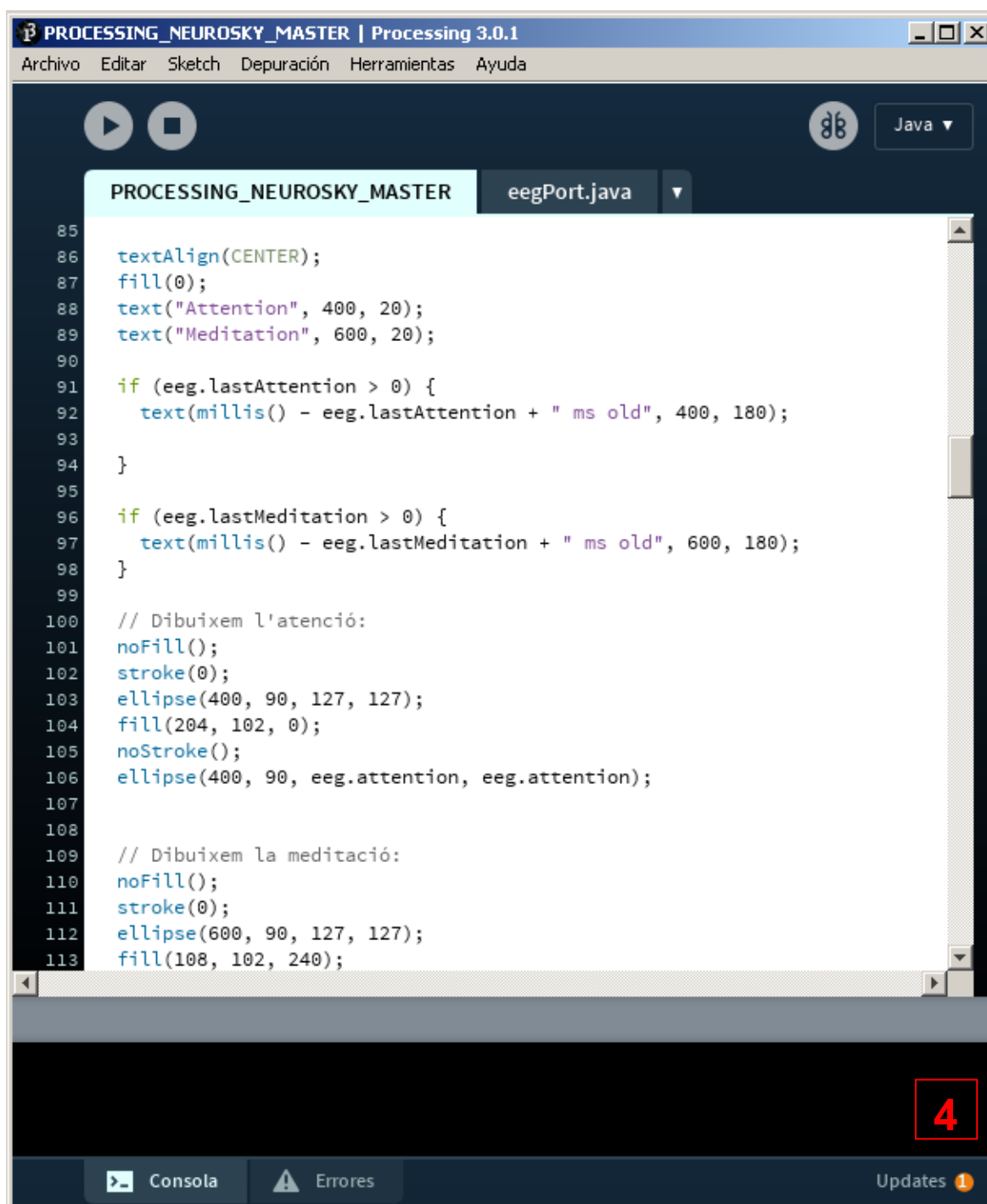
Aquestes acotacions, que en el funcionament intern dels programes no tenen gens de transcendència, sí que poden ser importants per poder comprendre l'estructura i el funcionament de cada un dels programes emprats en aquest treball.

## 5.1.1 PROCESSING NEUROSKY MASTER









PROCESSING\_NEUROSKY\_MASTER | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

Java ▾

PROCESSING\_NEUROSKY\_MASTER eegPort.java ▾

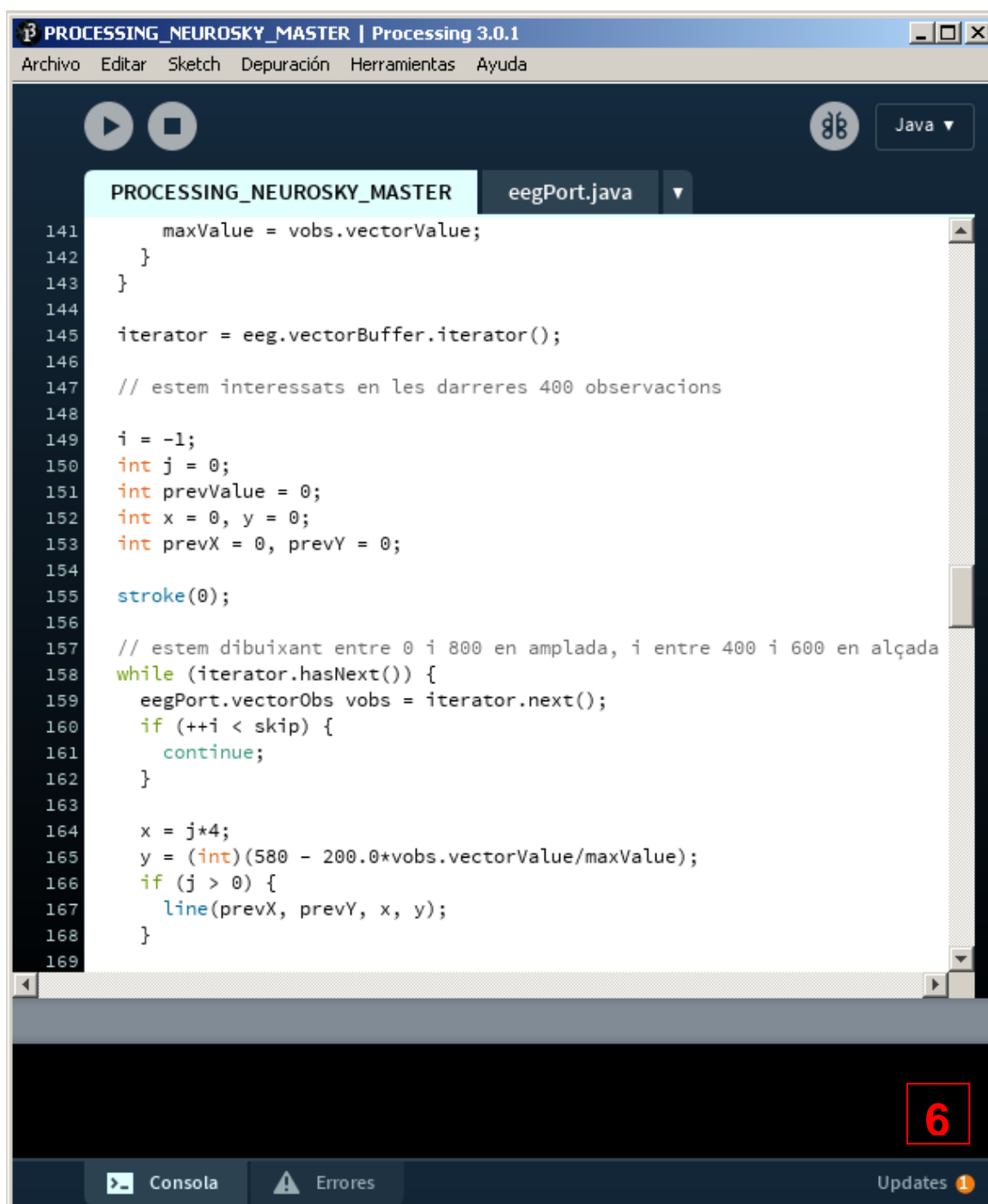
```

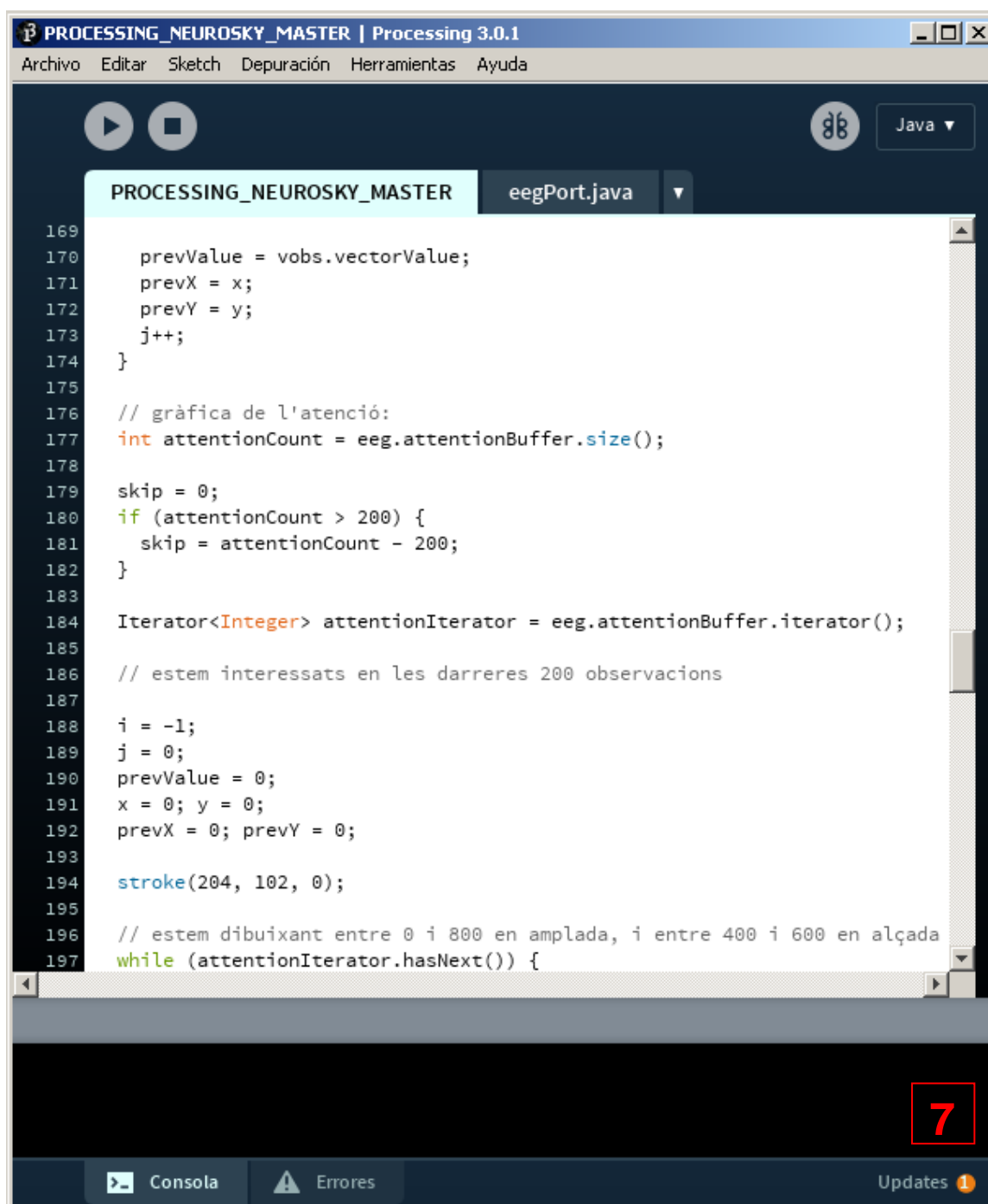
113 fill(108, 102, 240);
114 noStroke();
115 ellipse(600, 90, eeg.meditation, eeg.meditation);
116
117 // Dibuixa senyal
118
119
120 // Valors del vector de la gràfica:
121 // primer obtenim el valor màxim
122 int maxValue = 0;
123 Iterator<eegPort.vectorObs> iterator;
124 iterator = eeg.vectorBuffer.iterator();
125 int vectorCount = eeg.vectorBuffer.size();
126
127 int skip = 0;
128 if (vectorCount > 200) {
129     skip = vectorCount - 200;
130 }
131
132 int i = -1;
133
134 while (iterator.hasNext()) {
135     eegPort.vectorObs vobs = iterator.next();
136     if (++i < skip) {
137         continue;
138     }
139
140     if (vobs.vectorValue > maxValue) {
141         maxValue = vobs.vectorValue;

```

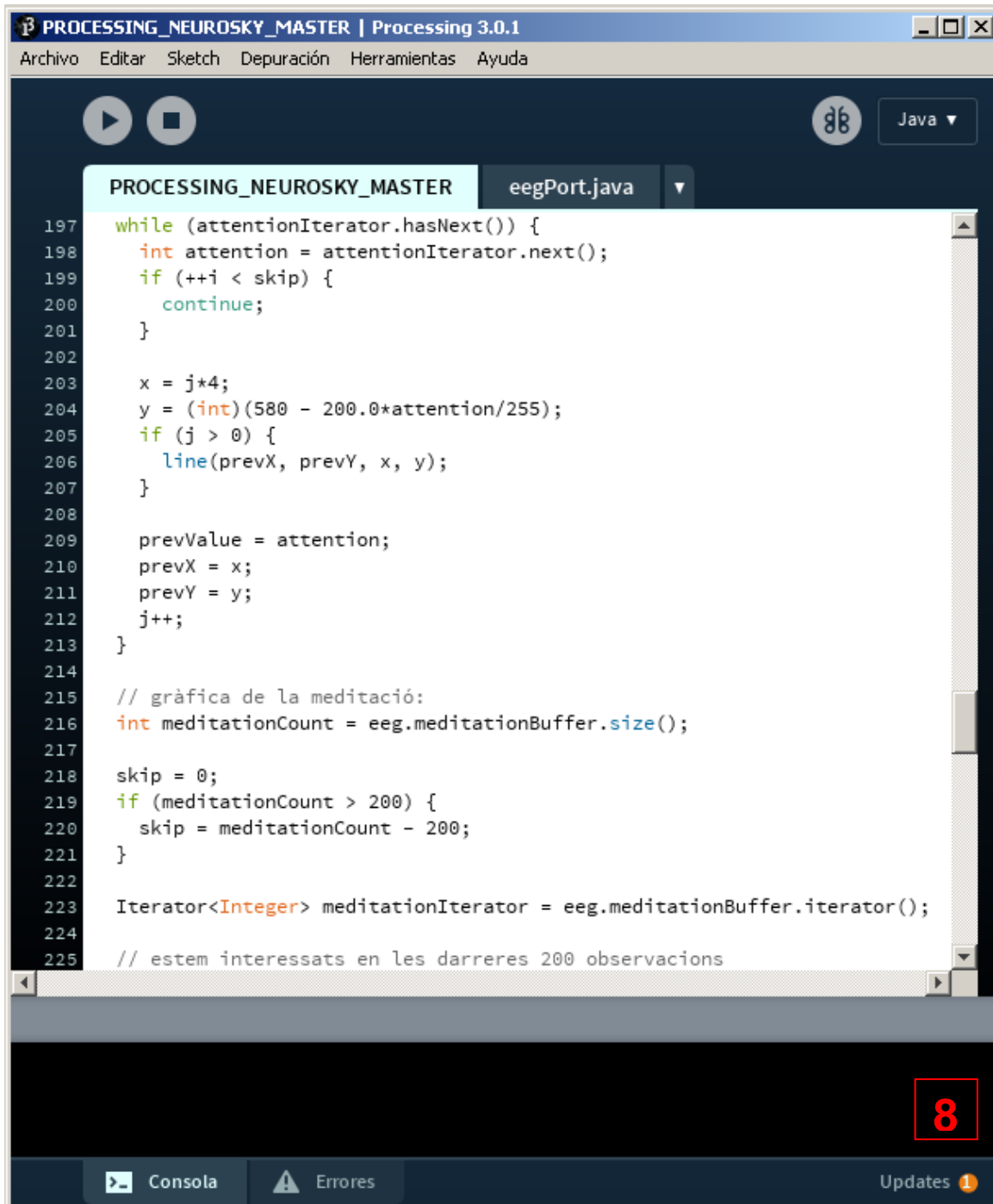
5

Consola Errores Updates 1









PROCESSING\_NEUROSKY\_MASTER | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

Java ▾

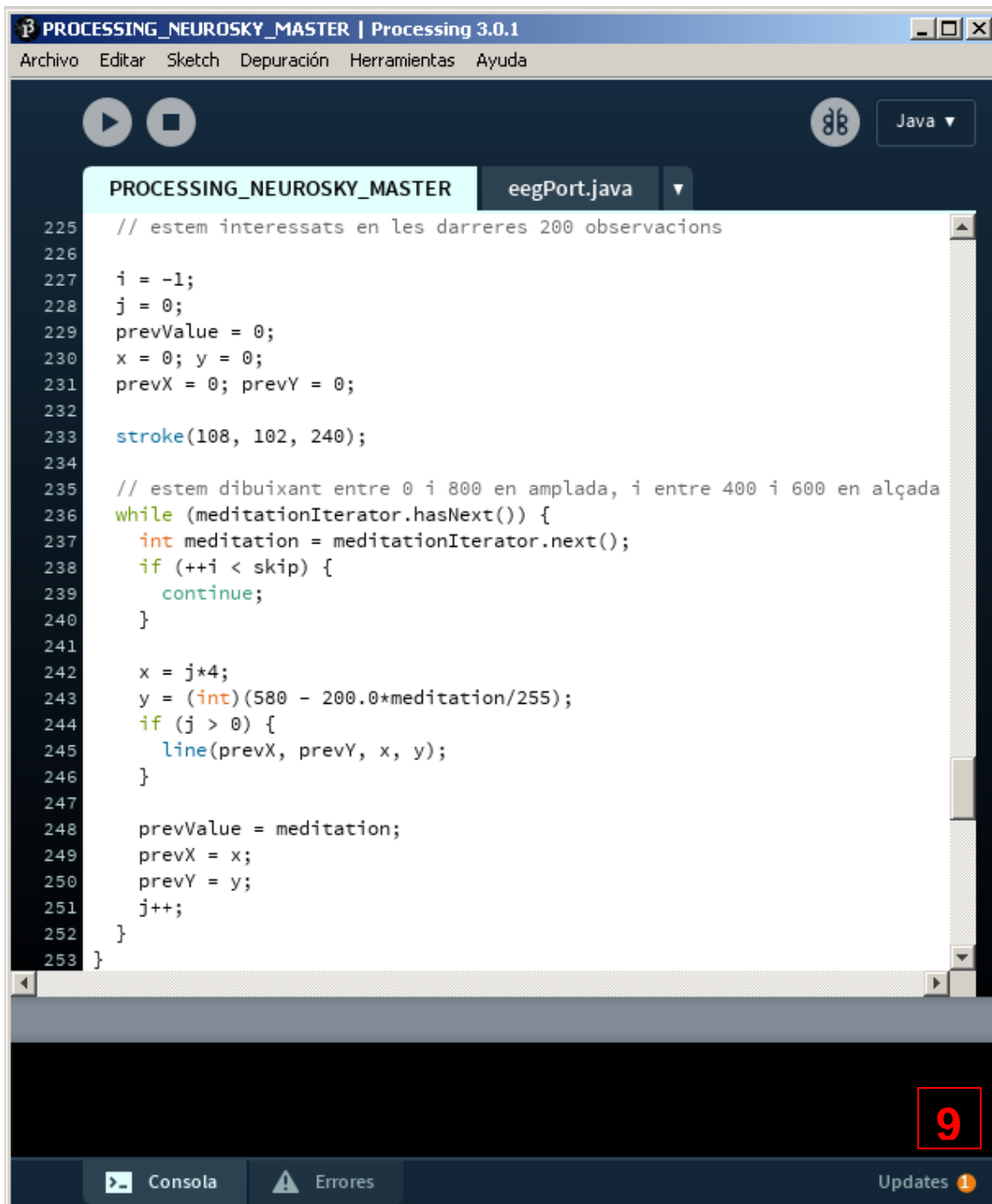
PROCESSING\_NEUROSKY\_MASTER eegPort.java ▾

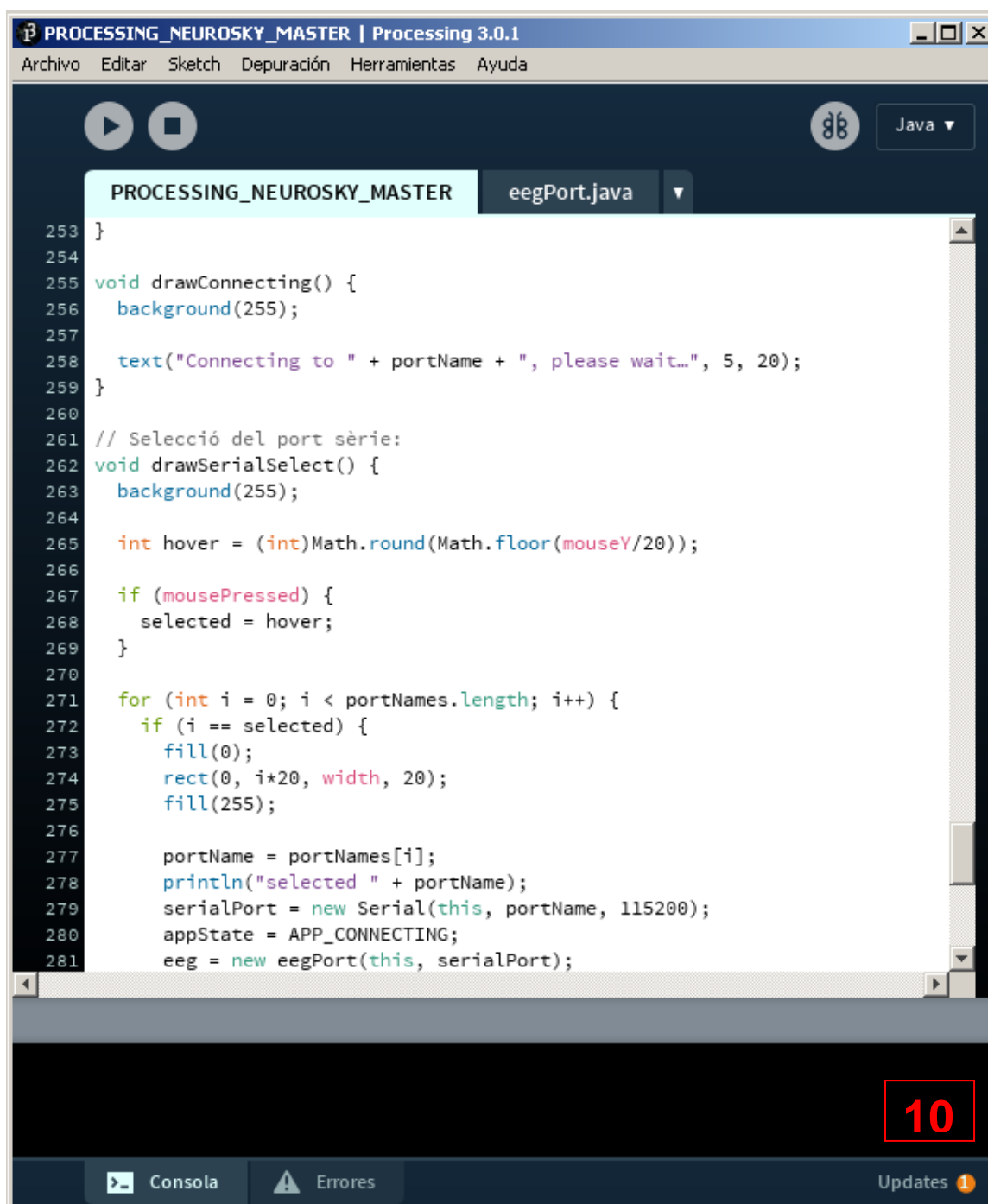
```

197 while (attentionIterator.hasNext()) {
198     int attention = attentionIterator.next();
199     if (++i < skip) {
200         continue;
201     }
202
203     x = j*4;
204     y = (int)(580 - 200.0*attention/255);
205     if (j > 0) {
206         line(prevX, prevY, x, y);
207     }
208
209     prevValue = attention;
210     prevX = x;
211     prevY = y;
212     j++;
213 }
214
215 // gràfica de la meditació:
216 int meditationCount = eeg.meditationBuffer.size();
217
218 skip = 0;
219 if (meditationCount > 200) {
220     skip = meditationCount - 200;
221 }
222
223 Iterator<Integer> meditationIterator = eeg.meditationBuffer.iterator();
224
225 // estem interessats en les darreres 200 observacions
    
```

8

Consola Errores Updates 1





PROCESSING\_NEUROSKY\_MASTER | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

Java ▾

PROCESSING\_NEUROSKY\_MASTER eegPort.java ▾

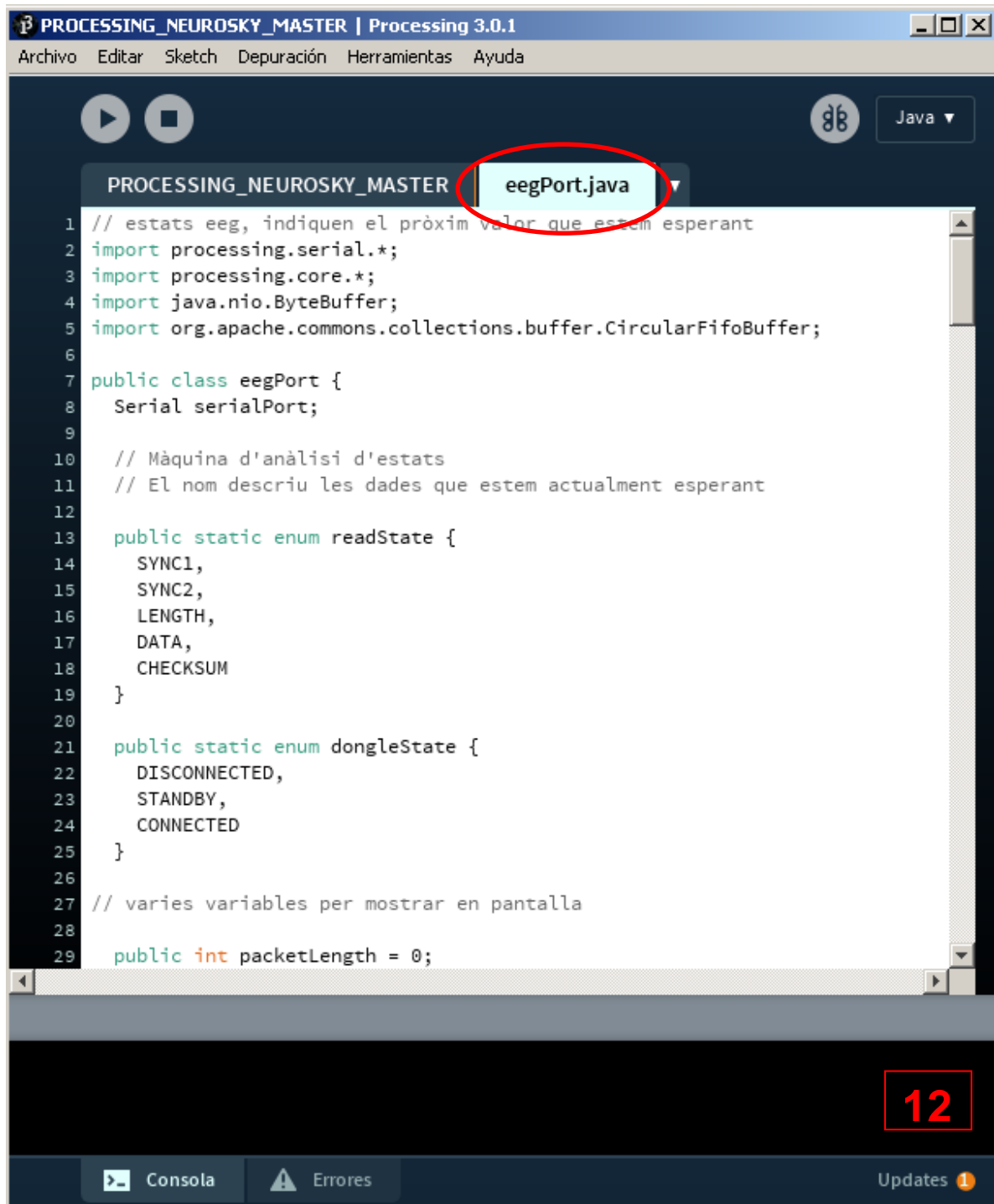
```

281     eeg = new eegPort(this, serialPort);
282     delay(500);
283     eeg.refresh();
284   } else if (i == hover) {
285     fill(200, 200, 240);
286     noStroke();
287     rect(0, i*20, width, 20);
288     fill(0);
289   } else {
290     fill(0);
291   }
292   text(portNames[i], 5, (i+1)*20);
293 }
294 }
295
296 void serialEvent(Serial p) {
297   while (p.available() > 0) {
298     int inByte = p.read();
299     eeg.serialByte(inByte);
300     if (inByte == 170 && appState < APP_CONNECTED) {
301       println("Connected");
302       appState = APP_CONNECTED;
303       frameRate(10);
304     }
305   }
306 }
307
308
309

```

11

Consola Errores Updates 1



Atenció! Aquí comença el codi d'un programa adjunt, que també forma part del *Processing NeuroSky Master*. Aquest programa adjunt és l'encarregat d'emmagatzemar i organitzar les variables amb les quals treballa el *Processing NeuroSky Master*.

PROCESSING\_NEUROSKY\_MASTER | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

PROCESSING\_NEUROSKY\_MASTER eegPort.java

```

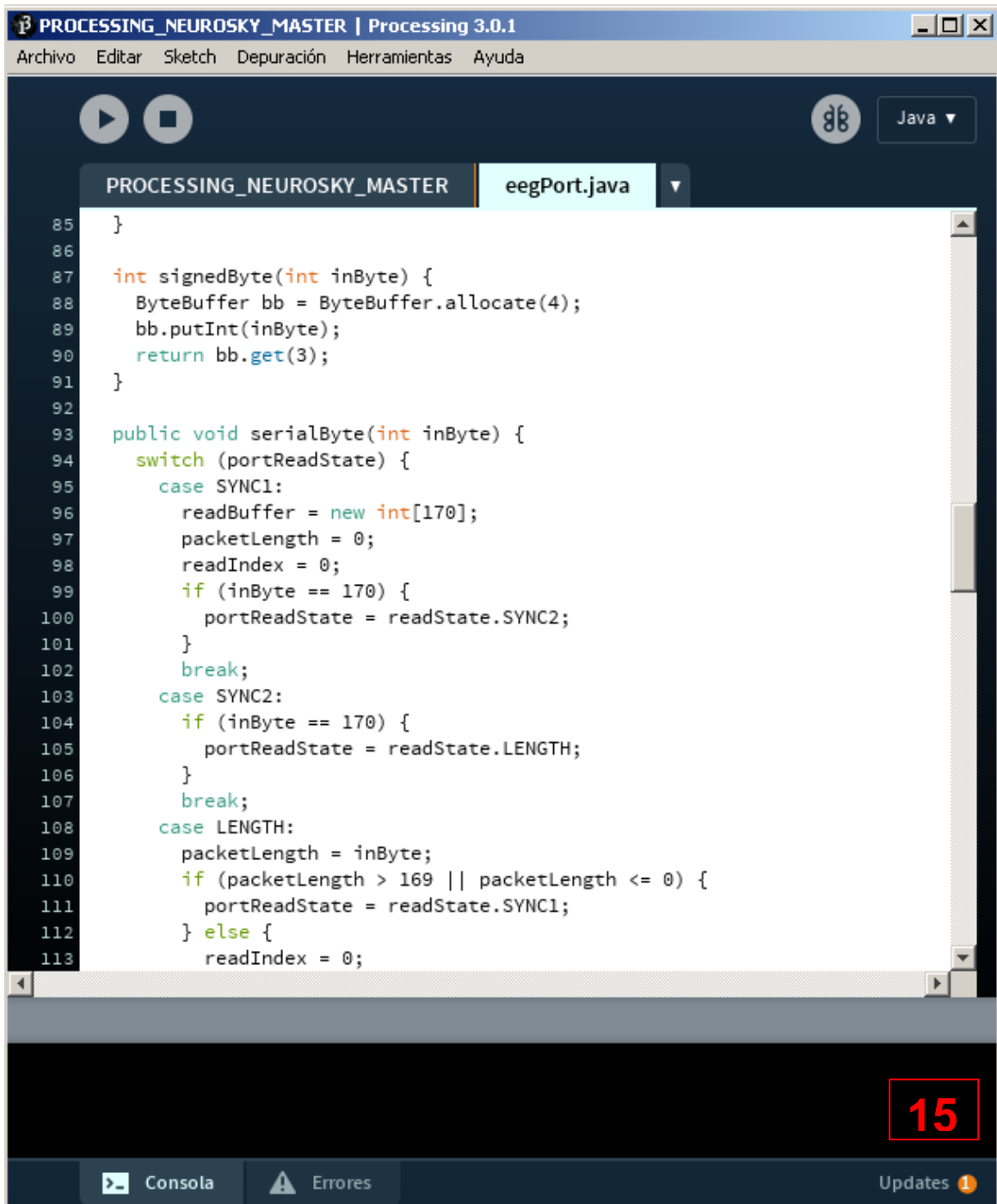
29 public int packetLength = 0;
30 public int packetCode = 0;
31 public int poorSignal = 255;
32 public int readIndex = 0;
33 public int attention = 0;
34 public int meditation = 0;
35 public int lastEvent = 0;
36 public int lastAttention = 0;
37 public int lastMeditation = 0;
38 int vectorBytesLeft = 0;
39 public dongleState portDongleState = dongleState.DISCONNECTED;
40 public readState portReadState = readState.SYNC1;
41
42 public int rawSequence = 0;
43 public int vectorSequence = 0;
44
45 public int failedChecksumCount = 0;
46
47 int readBuffer[];
48
49 CircularFifoBuffer rawDataBuffer;
50 CircularFifoBuffer vectorBuffer;
51 CircularFifoBuffer attentionBuffer;
52 CircularFifoBuffer meditationBuffer;
53
54 //reserva pels valors originals
55
56 PApplet app;
57

```

13

Consola Errores Updates 1





PROCESSING\_NEUROSKY\_MASTER | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

Java ▾

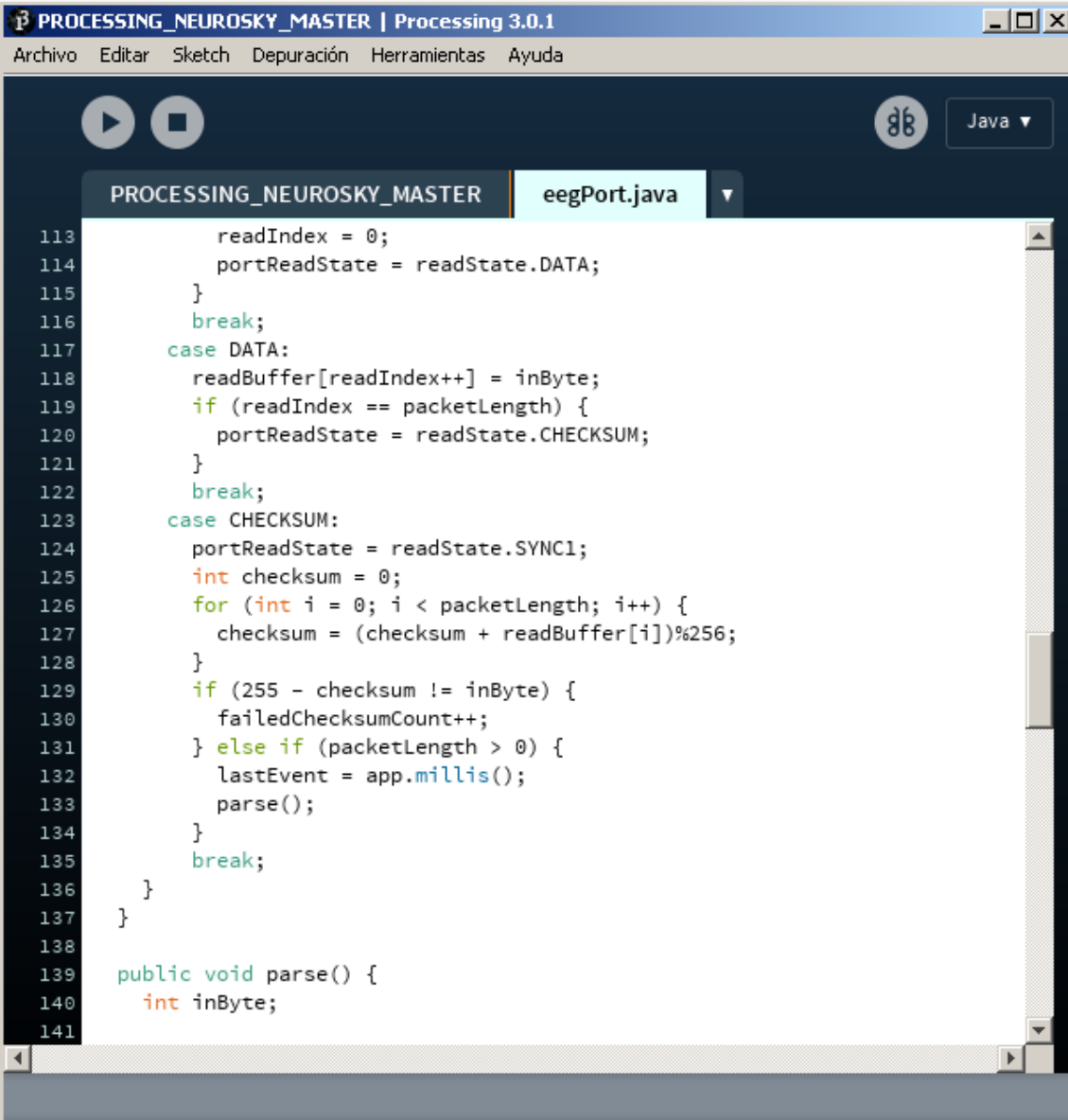
PROCESSING\_NEUROSKY\_MASTER eegPort.java ▾

```
85 }
86
87 int signedByte(int inByte) {
88     ByteBuffer bb = ByteBuffer.allocate(4);
89     bb.putInt(inByte);
90     return bb.get(3);
91 }
92
93 public void serialByte(int inByte) {
94     switch (portReadState) {
95         case SYNC1:
96             readBuffer = new int[170];
97             packetLength = 0;
98             readIndex = 0;
99             if (inByte == 170) {
100                 portReadState = readState.SYNC2;
101             }
102             break;
103         case SYNC2:
104             if (inByte == 170) {
105                 portReadState = readState.LENGTH;
106             }
107             break;
108         case LENGTH:
109             packetLength = inByte;
110             if (packetLength > 169 || packetLength <= 0) {
111                 portReadState = readState.SYNC1;
112             } else {
113                 readIndex = 0;
```

15

Consola Errores Updates 1





```
113         readIndex = 0;
114         portReadState = readState.DATA;
115     }
116     break;
117 case DATA:
118     readBuffer[readIndex++] = inByte;
119     if (readIndex == packetLength) {
120         portReadState = readState.CHECKSUM;
121     }
122     break;
123 case CHECKSUM:
124     portReadState = readState.SYNC1;
125     int checksum = 0;
126     for (int i = 0; i < packetLength; i++) {
127         checksum = (checksum + readBuffer[i])%256;
128     }
129     if (255 - checksum != inByte) {
130         failedChecksumCount++;
131     } else if (packetLength > 0) {
132         lastEvent = app.millis();
133         parse();
134     }
135     break;
136 }
137 }
138
139 public void parse() {
140     int inByte;
141 }
```

16

```

141
142   for (int i = 0; i < packetLength; i++) {
143       switch(readBuffer[i]) {
144           case 212:          // "standby" o espera
145               portDongleState = dongleState.STANDBY;
146               break;
147           case 208:          // connectat
148               portDongleState = dongleState.CONNECTED;
149               break;
150           case 2:            // senyal pobre
151               inByte = readBuffer[++i];
152               if (inByte == 255 && poorSignal == 0) {
153                   inByte = 49;
154               } else if (inByte < 200 && poorSignal == 200) {
155                   inByte = 254;
156               }
157               poorSignal = inByte;
158               break;
159           case 4:            // atenció
160               inByte = signedByte(readBuffer[++i]);
161               if (inByte > 0) {
162                   attention = inByte;
163               }
164
165               lastAttention = app.millis();
166               attentionBuffer.add(attention);
167               break;
168           case 5:            // meditació
169               inByte = signedByte(readBuffer[++i]);

```

17

PROCESSING\_NEUROSKY\_MASTER | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

PROCESSING\_NEUROSKY\_MASTER eegPort.java

```

169         inByte = signedByte(readBuffer[++i]);
170         if (inByte > 0) {
171             meditation = inByte;
172         }
173
174         lastMeditation = app.millis();
175         meditationBuffer.add(meditation);
176         break;
177     case 128: // valors originals
178         int rawRowLength = readBuffer[++i];
179         int rawA = readBuffer[++i];
180         int rawB = readBuffer[++i];
181
182         ByteBuffer bbA = ByteBuffer.allocate(4);
183         bbA.putInt(rawA);
184         ByteBuffer bbB = ByteBuffer.allocate(4);
185         bbB.putInt(rawB);
186
187         ByteBuffer bb = ByteBuffer.allocate(2);
188
189         bb.put(1, bbA.get(3));
190         bb.put(0, bbB.get(3));
191
192         short rawValue = bb.getShort(0);
193
194         rawSequence++;
195         rawObs obs = new rawObs();
196         obs.sequence = rawSequence;
197         obs.timestamp = app.millis();

```

18

Consola Errores Updates 1

```

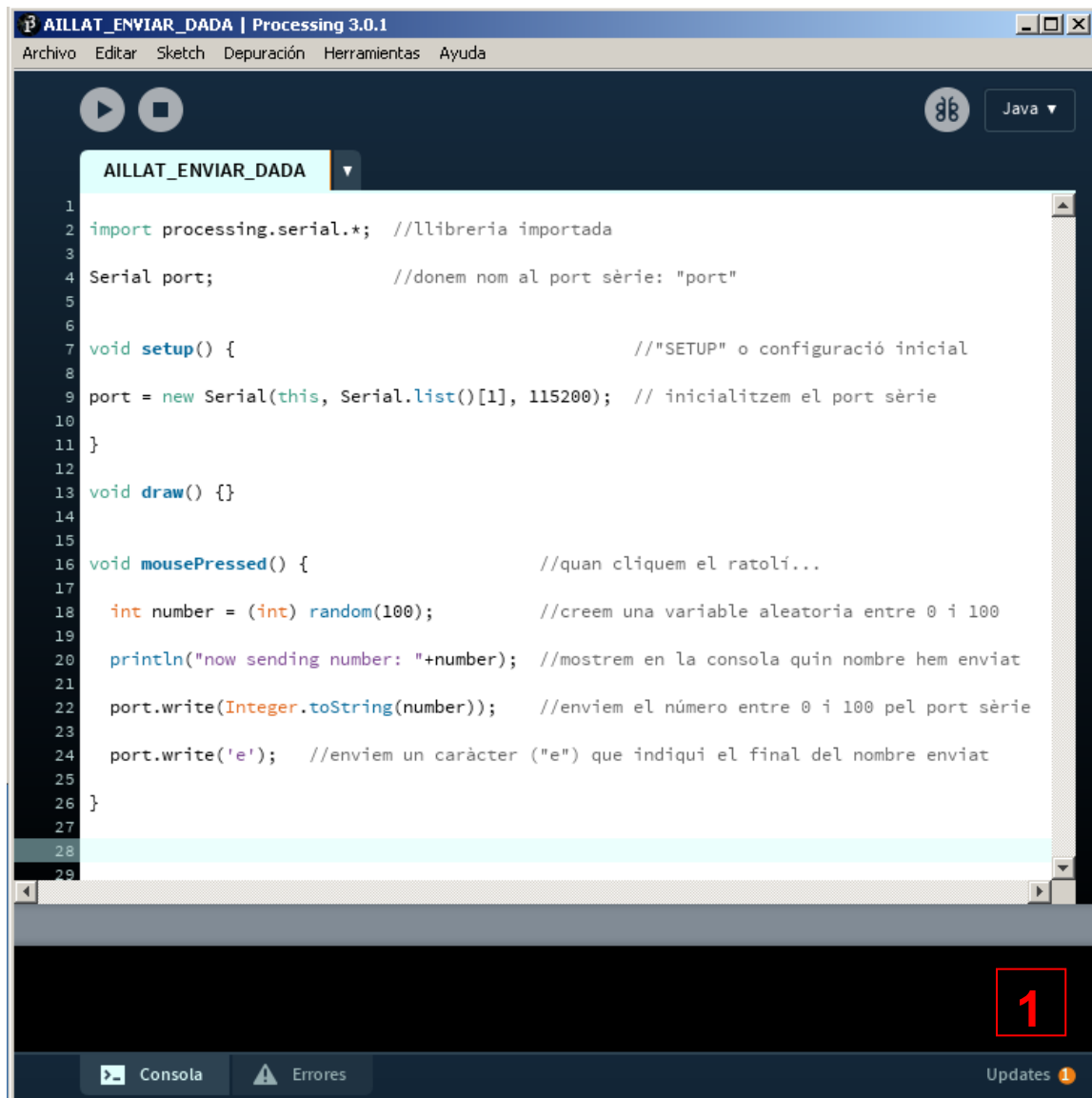
197     obs.timestamp = app.millis();
198     obs.rawValue = rawValue;
199
200     rawDataBuffer.add(obs);
201     break;
202 case 131:      // vector
203     int vectorLength = readBuffer[++i];
204     if (vectorLength != 24) {
205         // si dóna algun error
206         break;
207     }
208
209     for (int j = 0; j < 8; j++) {
210         int vecA = readBuffer[++i];
211         int vecB = readBuffer[++i];
212         int vecC = readBuffer[++i];
213         vectorObs vobs = new vectorObs();
214         vobs.timestamp = app.millis();
215         vobs.sequence = ++vectorSequence;
216         vobs.vectorValue = vecA*255*255 + vecB*255 + vecC;
217         vectorBuffer.add(vobs);
218     }
219     default:    // desconegut
220
221     break;
222 }
223 }
224 }
225 }
    
```

19


Consola   Errores   Updates 1

## 5.1.2 PROGRAMES AÏLLATS

### 5.1.2.1 PROCESSING



## 5.1.2.2 ARDUINO



```

AILLAT_REBRE_DADA Arduino 1.6.5
Archivo  Editar  Programa  Herramientas  Ayuda

int c; //definim les variables
int atencio=0;
int atenciomapejada;

void setup() { //inici del "SETUP" o configuració inicial

  Serial.begin(115200); //posem en marxa el port sèrie, per on arribaran les dades
  pinMode(9,OUTPUT); //configurem el pin 9 en mode "sortida" (OUTPUT)

} //final del "SETUP" o configuració inicial

void loop() {

  while (Serial.available()) { //*****
    c = Serial.read(); //*****
    if ((c >= '0') && (c <= '9')) { //*****
      atencio = 10 * atencio + c - '0'; //*****

    atenciomapejada=map(atencio,0,100,0,255); //mapegem*** l'atenció: de 0-100 a 0-255

    analogWrite(9, atenciomapejada); //donem al LED + o - voltatge en funció de l'atenció

  }
  else if (c == 'e') { //*****
    atencio = 0; //*****
  }
}
}

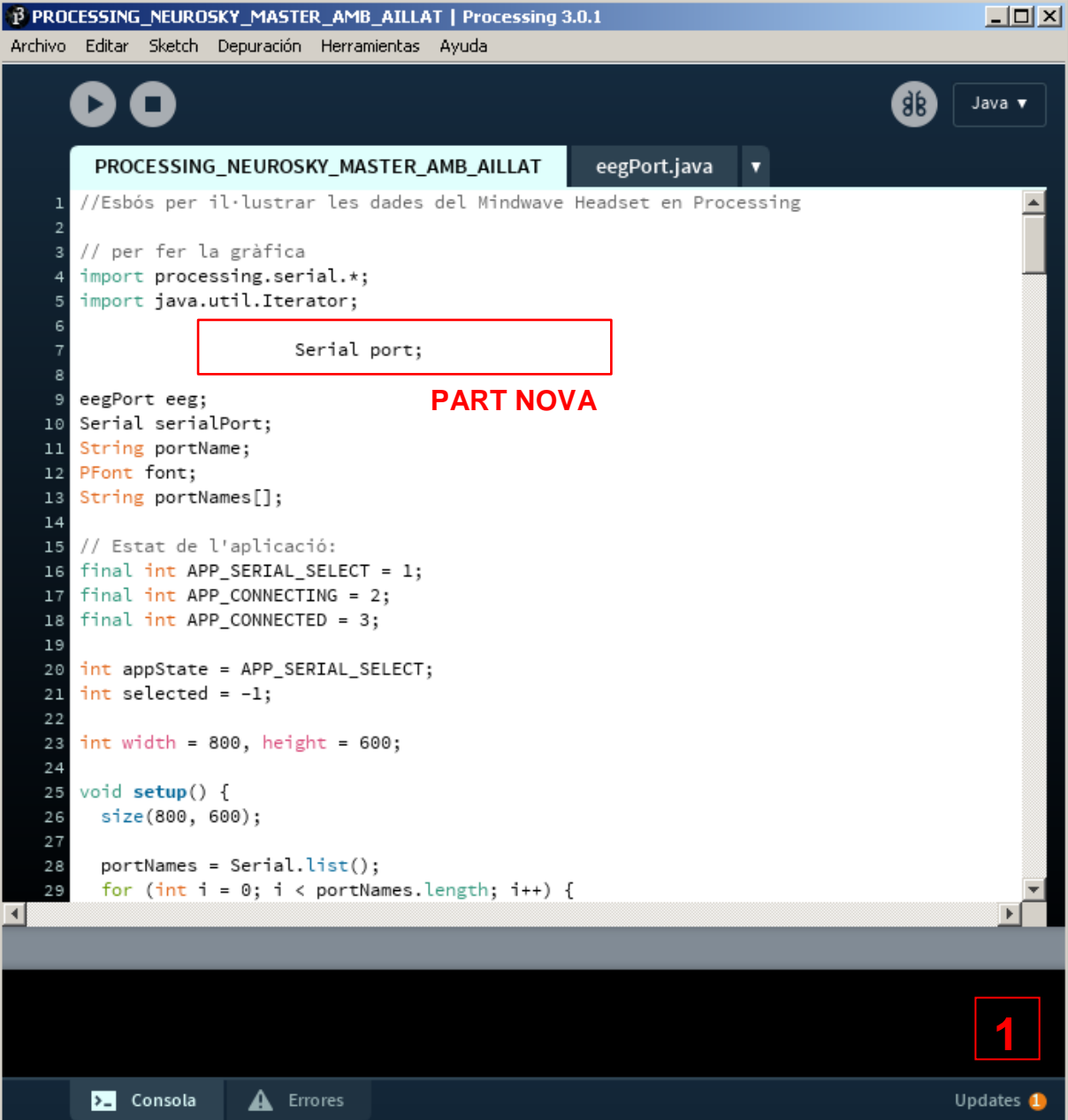
```

29 Arduino/Genuino Uno on COM6

\*\*\*\*\*: El programa llegeix les dades que li arriben pel port sèrie. Ara bé, les dades (un número entre 0 i 100) no poden ser llegides directament, ja que són enviades xifra a xifra. Per exemple, en enviar el nombre 83, el programa d'entorn Processing envia primer el 8 i després el 3. Els comandaments assenyalats amb els asteriscs són els encarregats de rebre aquesta dada "fraccionada" i convertir-la de nou en un número de 0 a 100.

\*\*\*: *mapejar* un nombre significa **extrapolar-lo**. En aquest cas, per exemple el 0 seguiria sent un 0, però el 100 passaria a ser un 255, tal com el 50 passaria a ser el 127'5, o el 20 el 51.

## 5.1.3 PROCESSING NEUROSKY MASTER + PROGRAMA AÏLLAT



```

PROCESSING_NEUROSKY_MASTER_AMB_AILLAT | Processing 3.0.1
Archivo  Editar  Sketch  Depuración  Herramientas  Ayuda

PROCESSING_NEUROSKY_MASTER_AMB_AILLAT  eegPort.java

1 //Esbós per il·lustrar les dades del Mindwave Headset en Processing
2
3 // per fer la gràfica
4 import processing.serial.*;
5 import java.util.Iterator;
6
7 Serial port;
8
9 eegPort eeg;
10 Serial serialPort;
11 String portName;
12 PFont font;
13 String portNames[];
14
15 // Estat de l'aplicació:
16 final int APP_SERIAL_SELECT = 1;
17 final int APP_CONNECTING = 2;
18 final int APP_CONNECTED = 3;
19
20 int appState = APP_SERIAL_SELECT;
21 int selected = -1;
22
23 int width = 800, height = 600;
24
25 void setup() {
26   size(800, 600);
27
28   portNames = Serial.list();
29   for (int i = 0; i < portNames.length; i++) {

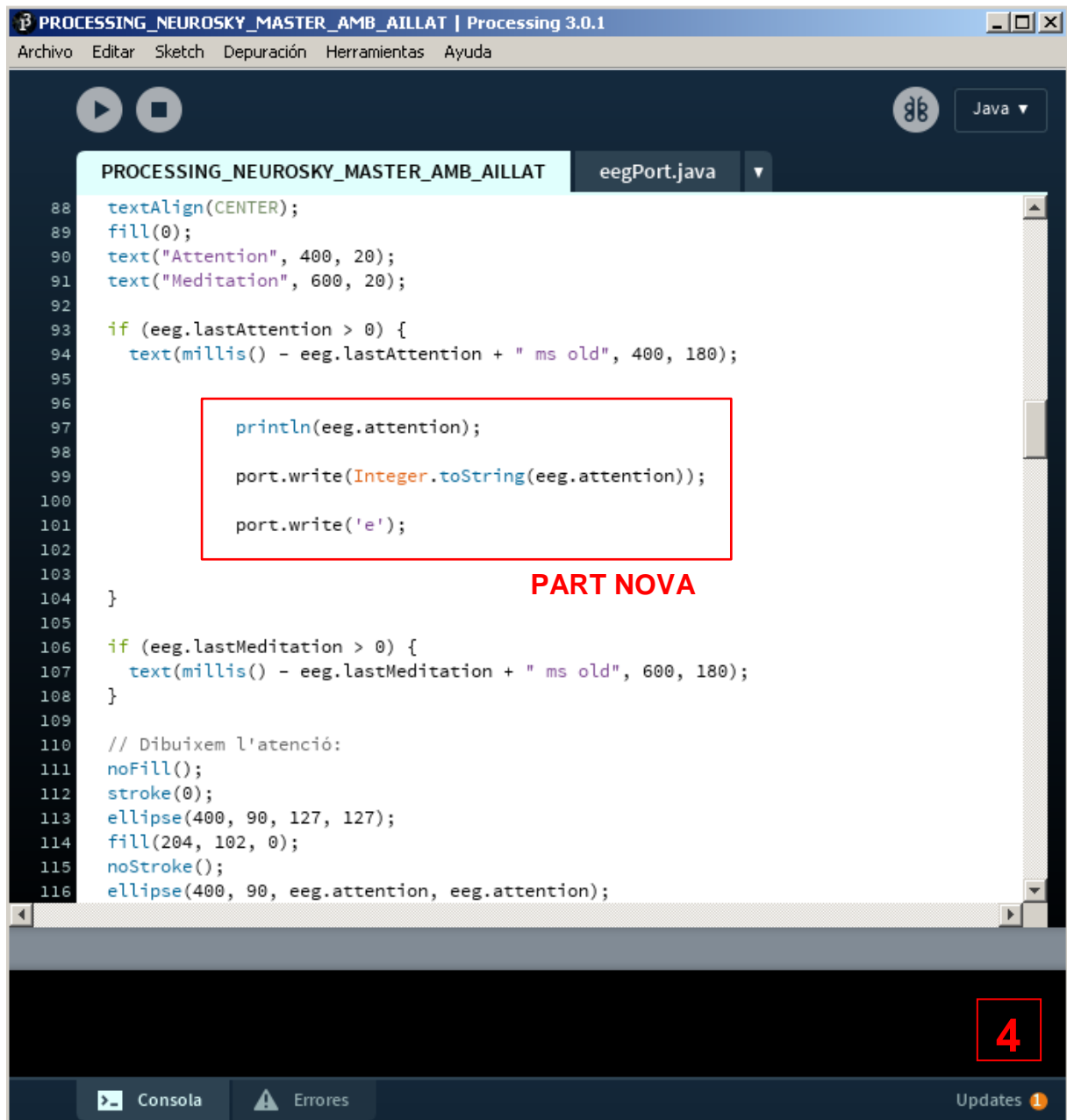
```

**PART NOVA**

**1**

Consola Errores Updates

Aquesta instrucció, situada a l'inici del programa, dóna nom (*port*) al port sèrie pel qual enviarem les dades cap a la tauleta Arduino.



Aquestes 3 instruccions estan situades de manera que s'executin quan el nivell d'atenció sigui superior a 0, tal com es pot veure en la línia 93. Per ordre, el que fan és:

- Mostrar la dada *atenció* a la consola de l'entorn Processing
- Enviar la dada *atenció* pel port sèrie (recordem, xifra a xifra)
- Enviar un caràcter (e) per indicar que ja hem acabat d'enviar la variable (el programa en Arduino reconstruirà un altre cop la variable).



PROCESSING\_NEUROSKEY\_MASTER\_AMB\_AILLAT | Processing 3.0.1

Archivo Editar Sketch Depuración Herramientas Ayuda

Java ▾

PROCESSING\_NEUROSKEY\_MASTER\_AMB\_AILLAT eegPort.java ▾

```

283 fill(0);
284 rect(0, i*20, width, 20);
285 fill(255);
286
287 portName = portNames[i];
288 println("selected " + portName);
289 serialPort = new Serial(this, portName, 115200);
290
291 port = new Serial(this, Serial.list()[2], 115200);
292
293 appState = APP_CONNECTING;
294 eeg = new eegPort(this, serialPort);
295 delay(500);
296 eeg.refresh();
297 } else if (i == hover) {
298 fill(200, 200, 240);
299 noStroke();
300 rect(0, i*20, width, 20);
301 fill(0);
302 } else {
303 fill(0);
304 }
305 text(portNames[i], 5, (i+1)*20);
306 }
307 }
308
309 void serialEvent(Serial p) {
310 while (p.available() > 0) {
311 int inByte = p.read();

```

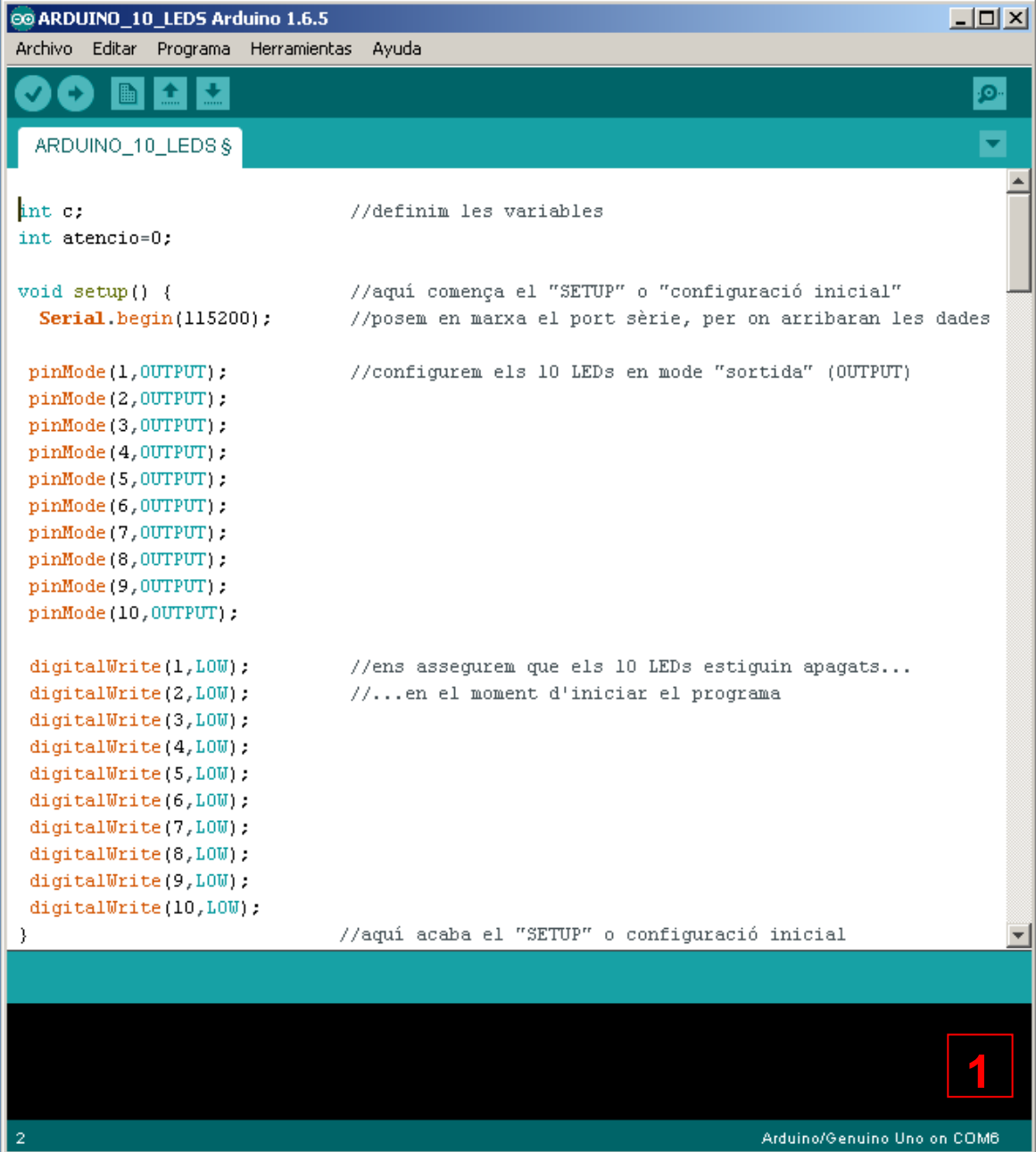
**PART NOVA**

**11**

Consola Errores Updates 1

Aquesta instrucció obra el port sèrie que abans havíem declarat com a *port*, per on s'enviaran les dades cap a Arduino. Aquesta instrucció, tot i estar situada al final del programa, s'executa ben a l'inici, abans de començar a registrar les dades que detecta el casc.

## 5.1.4 ARDUINO 10 LEDS



```

ARDUINO_10_LEDS $

int c;                                //definim les variables
int atencio=0;

void setup() {                        //aquí comença el "SETUP" o "configuració inicial"
  Serial.begin(115200);               //posem en marxa el port sèrie, per on arribaran les dades

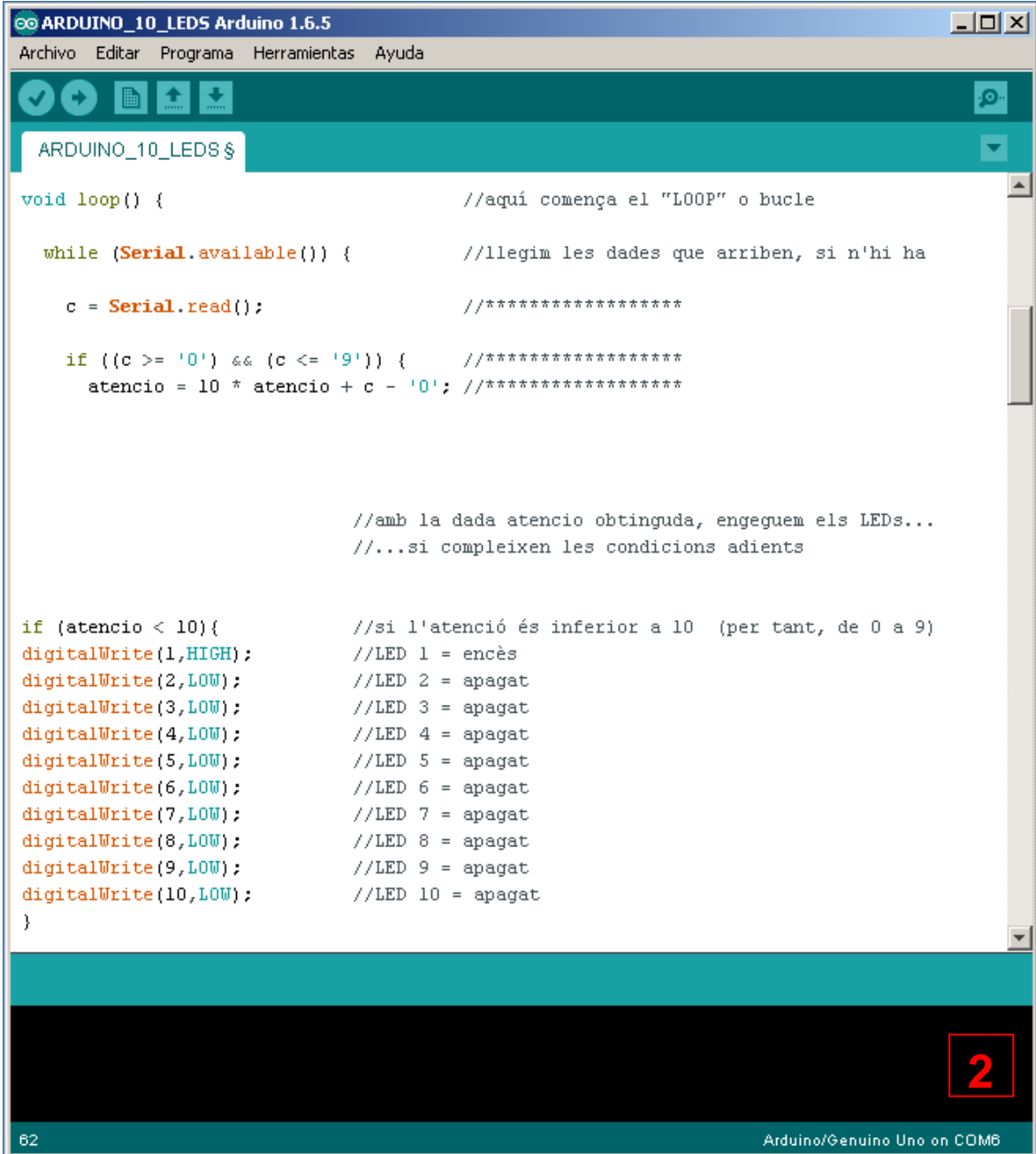
  pinMode(1,OUTPUT);                 //configurem els 10 LEDs en mode "sortida" (OUTPUT)
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);

  digitalWrite(1,LOW);               //ens assegurem que els 10 LEDs estiguin apagats...
  digitalWrite(2,LOW);               //...en el moment d'iniciar el programa
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);

}                                     //aquí acaba el "SETUP" o configuració inicial
  
```

1

2 Arduino/Genuino Uno on COM5



```

ARDUINO_10_LEDS Arduino 1.6.5
Archivo  Editor  Programa  Herramientas  Ayuda

void loop() {                                     //aquí comença el "LOOP" o bucle

  while (Serial.available()) {                   //llegim les dades que arriben, si n'hi ha

    c = Serial.read();                           //*****

    if ((c >= '0') && (c <= '9')) {               //*****
      atencio = 10 * atencio + c - '0';          //*****

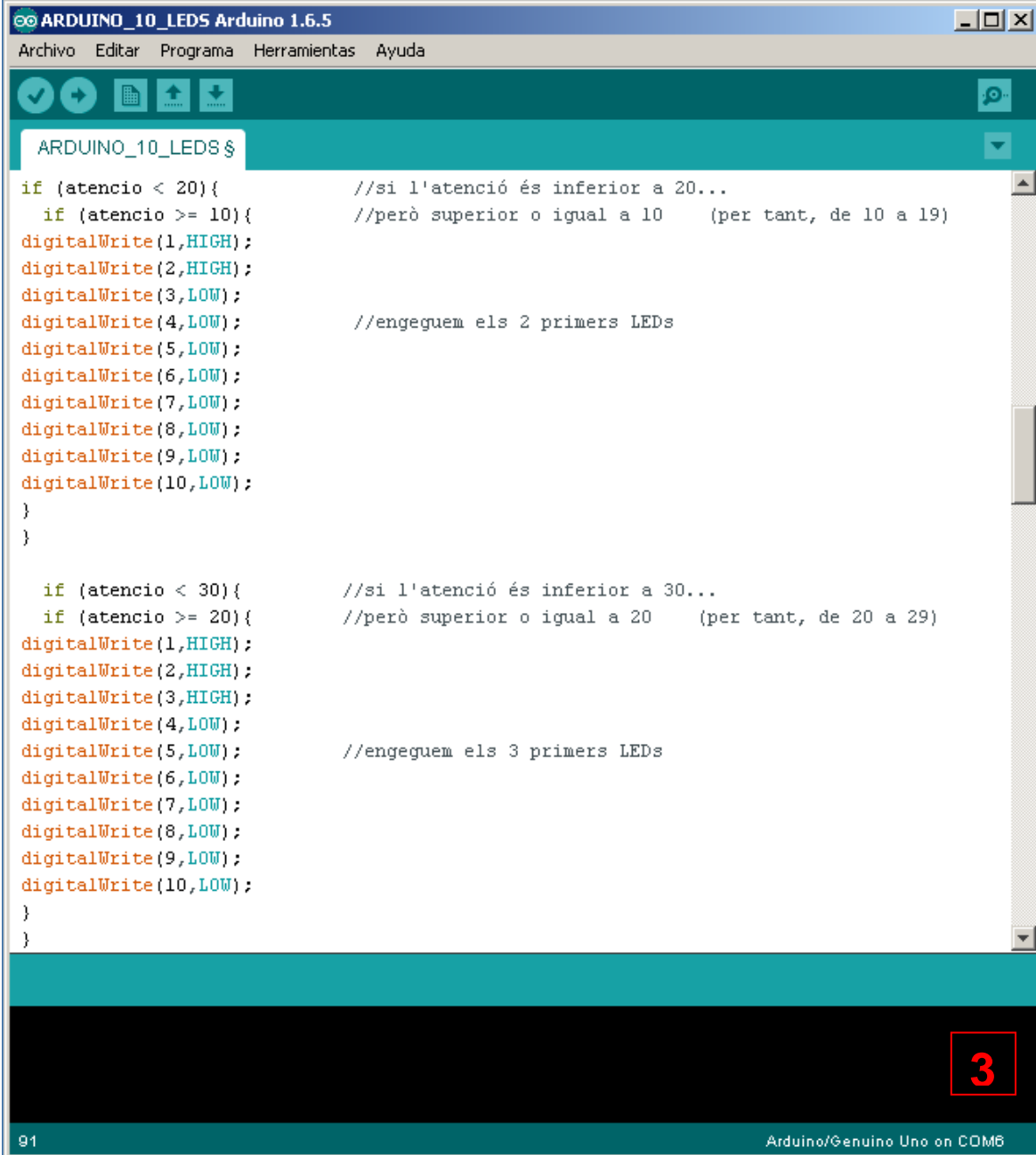
                                              //amb la dada atencio obtinguda, engegarem els LEDs...
                                              //...si compleixen les condicions adients

    if (atencio < 10){                           //si l'atenció és inferior a 10 (per tant, de 0 a 9)
      digitalWrite(1,HIGH);                      //LED 1 = encès
      digitalWrite(2,LOW);                       //LED 2 = apagat
      digitalWrite(3,LOW);                       //LED 3 = apagat
      digitalWrite(4,LOW);                       //LED 4 = apagat
      digitalWrite(5,LOW);                       //LED 5 = apagat
      digitalWrite(6,LOW);                       //LED 6 = apagat
      digitalWrite(7,LOW);                       //LED 7 = apagat
      digitalWrite(8,LOW);                       //LED 8 = apagat
      digitalWrite(9,LOW);                       //LED 9 = apagat
      digitalWrite(10,LOW);                      //LED 10 = apagat
    }
  }
}

```

62 Arduino/Genuino Uno on COM6

2



ARDUINO\_10\_LEDS Arduino 1.6.5

Archivo Editar Programa Herramientas Ayuda

ARDUINO\_10\_LEDS \$

```

if (atencio < 20){                //si l'atenció és inferior a 20...
    if (atencio >= 10){           //però superior o igual a 10    (per tant, de 10 a 19)
        digitalWrite(1,HIGH);
        digitalWrite(2,HIGH);
        digitalWrite(3,LOW);
        digitalWrite(4,LOW);      //engeguem els 2 primers LEDs
        digitalWrite(5,LOW);
        digitalWrite(6,LOW);
        digitalWrite(7,LOW);
        digitalWrite(8,LOW);
        digitalWrite(9,LOW);
        digitalWrite(10,LOW);
    }
}

    if (atencio < 30){            //si l'atenció és inferior a 30...
        if (atencio >= 20){       //però superior o igual a 20    (per tant, de 20 a 29)
            digitalWrite(1,HIGH);
            digitalWrite(2,HIGH);
            digitalWrite(3,HIGH);
            digitalWrite(4,LOW);
            digitalWrite(5,LOW);    //engeguem els 3 primers LEDs
            digitalWrite(6,LOW);
            digitalWrite(7,LOW);
            digitalWrite(8,LOW);
            digitalWrite(9,LOW);
            digitalWrite(10,LOW);
        }
    }
}

```

91

Arduino/Genuino Uno on COM6

3



```
ARDUINO_10_LEDS Arduino 1.6.5
Archivo  Editar  Programa  Herramientas  Ayuda

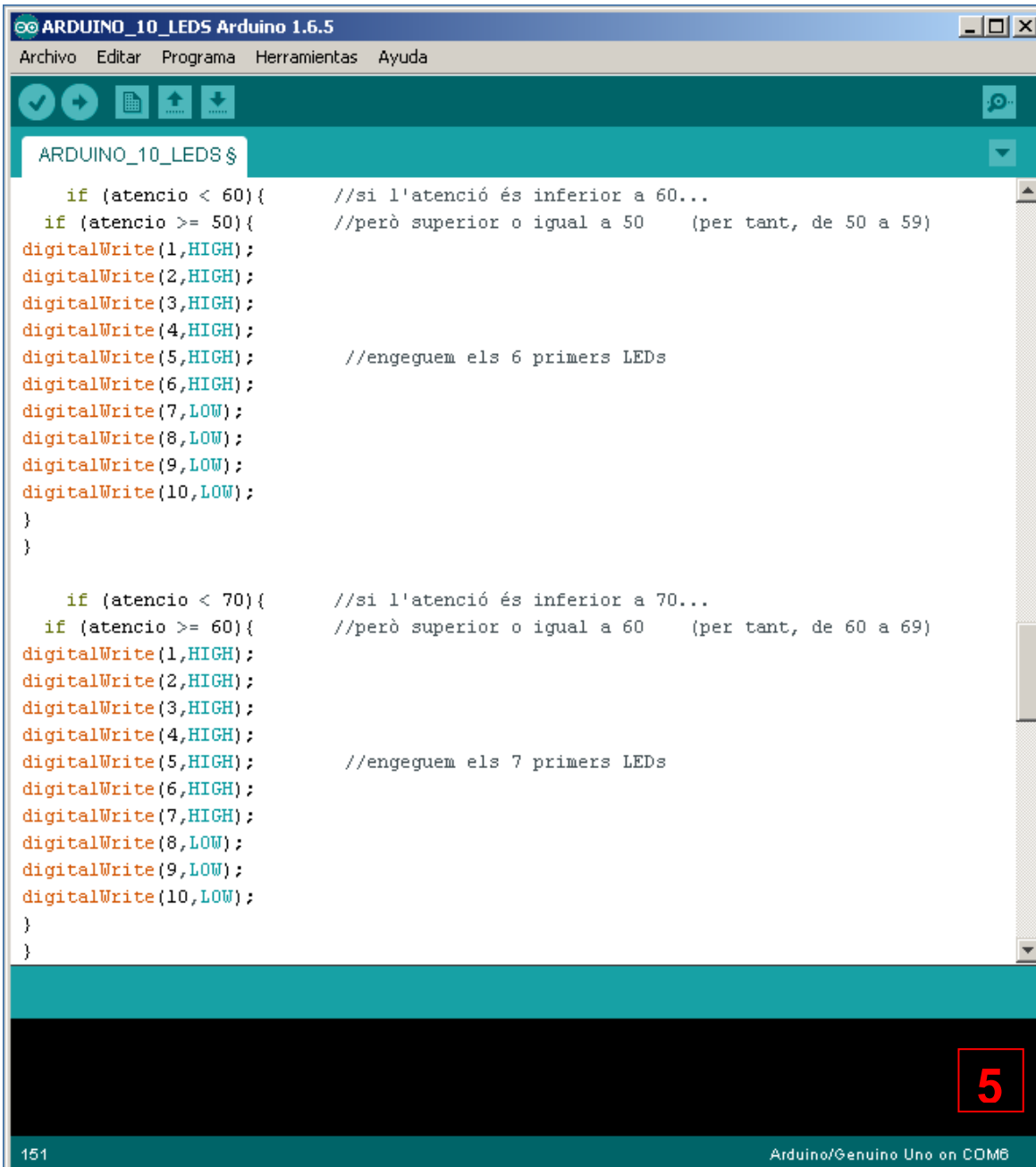
ARDUINO_10_LEDS $

    if (atencio < 40){          //si l'atenció és inferior a 40...
    if (atencio >= 30){          //però superior o igual a 30    (per tant, de 30 a 39)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,LOW);           //engeguem els 4 primers LEDs
digitalWrite(6,LOW);
digitalWrite(7,LOW);
digitalWrite(8,LOW);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
    }
    }

    if (atencio < 50){          //si l'atenció és inferior a 50...
    if (atencio >= 40){          //però superior o igual a 40    (per tant, de 40 a 49)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);          //engeguem els 5 primers LEDs
digitalWrite(6,LOW);
digitalWrite(7,LOW);
digitalWrite(8,LOW);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
    }
    }

121 Arduino/Genuino Uno on COM6
```

4



```

ARDUINO_10_LEDS $

    if (atencio < 60){          //si l'atenció és inferior a 60...
    if (atencio >= 50){          //però superior o igual a 50    (per tant, de 50 a 59)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);          //engeguem els 6 primers LEDs
digitalWrite(6,HIGH);
digitalWrite(7,LOW);
digitalWrite(8,LOW);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
    }
    }

    if (atencio < 70){          //si l'atenció és inferior a 70...
    if (atencio >= 60){          //però superior o igual a 60    (per tant, de 60 a 69)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);          //engeguem els 7 primers LEDs
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(8,LOW);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
    }
    }

```

151

Arduino/Genuino Uno on COM5

5

```

ARDUINO_10_LEDS Arduino 1.6.5
Archivo  Editar  Programa  Herramientas  Ayuda

[Icons] Salvar

ARDUINO_10_LEDS $

    if (atencio < 80){          //si l'atenció és inferior a 80...
    if (atencio >= 70){          //però superior o igual a 70      (per tant, de 70 a 79)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);          //engeguem els 8 primers LEDs
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(8,HIGH);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
    }
    }

    if (atencio < 90){          //si l'atenció és inferior a 90...
    if (atencio >= 80){          //però superior o igual a 80      (per tant, de 80 a 89)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);          //engeguem els 9 primers LEDs
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(8,HIGH);
digitalWrite(9,HIGH);
digitalWrite(10,LOW);
    }
    }

181 Arduino/Genuino Uno on COM6

```



```

ARDUINO_10_LEDS Arduino 1.6.5
Archivo  Editor  Programa  Herramientas  Ayuda

ARDUINO_10_LEDS $

    if (atencio <= 100){          //si l'atenció és inferior o igual a 100...
    if (atencio >= 90){           //però superior o igual a 90      (per tant, de 90 a 100)
digitalWrite(1,HIGH);
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);           //engeguem els 10 LEDs, tots
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(8,HIGH);
digitalWrite(9,HIGH);
digitalWrite(10,HIGH);

    }
    }

    else if (c == 'e') {         //*****

        atencio = 0;            //*****

    }

    }

}

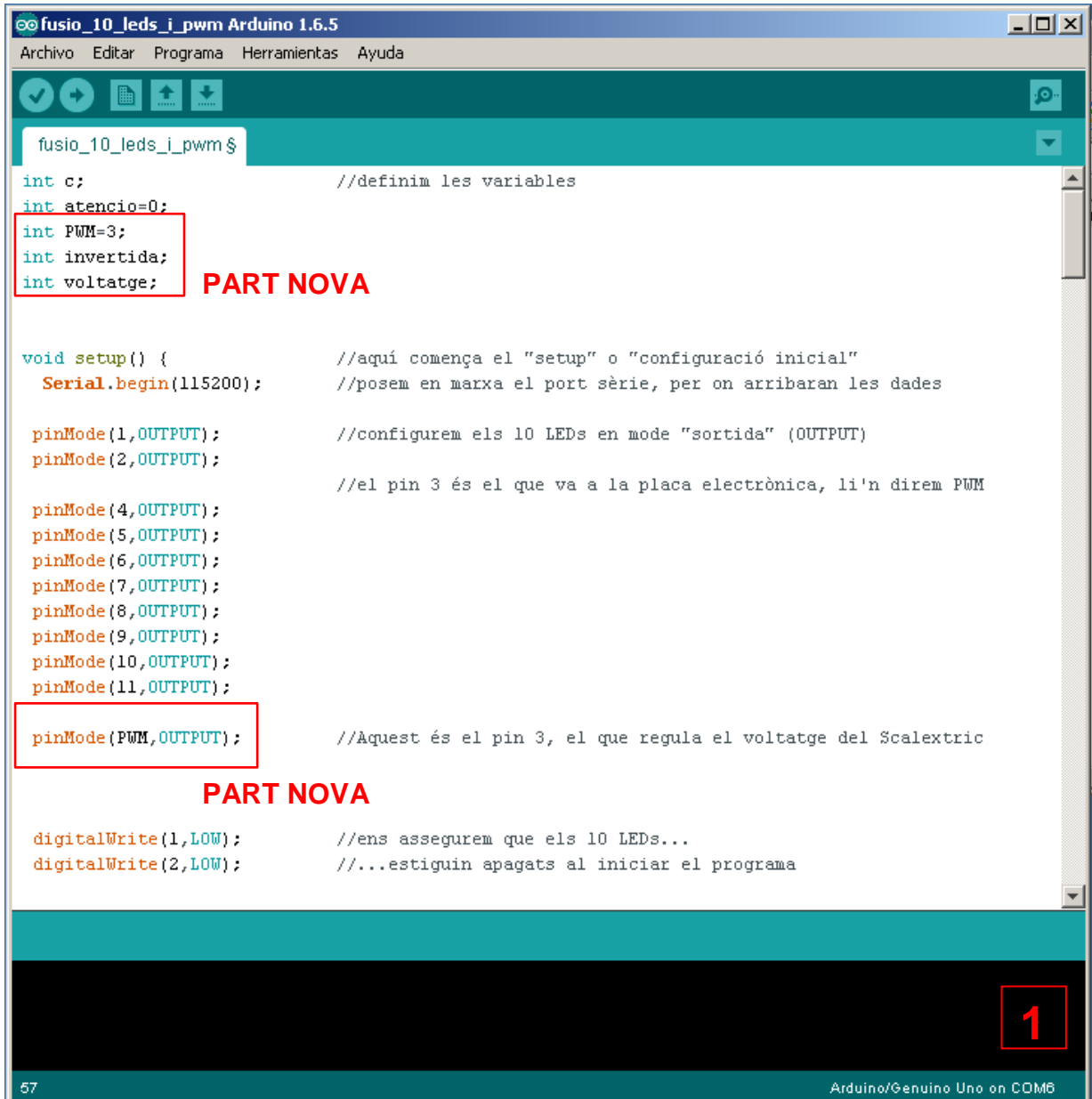
180                                     Arduino/Genuino Uno on COM8

```

\*\*\*\*\*: El programa, igual que succeïa en el programa aïllat per rebre les dades, llegeix les dades que li arriben pel port sèrie. Ara bé, les dades (un número entre 0 i 100) no poden ser llegides directament, ja que són enviades xifra a xifra. Per exemple, en enviar el nombre 83, el programa d'entorn Processing envia primer el 8 i després el 3. Els comandaments assenyalats amb els asteriscs són els encarregats de rebre aquesta dada "fraccionada" i convertir-la de nou en un número de 0 a 100.



## 5.1.5 ARDUINO→SCALEXTRIC



```

fusio_10_leds_i_pwm $
int c; //definim les variables
int atencio=0;
int PWM=3;
int invertida;
int voltatge;

void setup() { //aquí comença el "setup" o "configuració inicial"
  Serial.begin(115200); //posem en marxa el port sèrie, per on arribaran les dades

  pinMode(1,OUTPUT); //configurem els 10 LEDs en mode "sortida" (OUTPUT)
  pinMode(2,OUTPUT);
  //el pin 3 és el que va a la placa electrònica, li'n direm PWM

  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);

  pinMode(PWM,OUTPUT); //Aquest és el pin 3, el que regula el voltatge del Scalextric

  digitalWrite(1,LOW); //ens assegurem que els 10 LEDs...
  digitalWrite(2,LOW); //...estiguin apagats al iniciar el programa
  
```

En la incorporació del PWM al programa dels 10 LEDs s'ha produït una variació. La placa Arduino disposa de pocs pins digitals amb capacitat de produir una sortida PWM. Un d'ells és el pin 3. És per això que en aquest programa, el pin 3, que anomenarem PWM, serà el que ofereixi el voltatge variable a la placa electrònica, mentre que els encarregats d'encendre o no els LEDs seran els mateixos, amb l'excepció del pin 3, que serà substituït pel pin digital 11.

```

fusio_10_leds_i_pwm$
if ((c >= '0') && (c <= '9')) { //*****
  atencio = 10 * atencio + c - '0'; //*****

if (atencio<25){                //si l'atenció és inferior a 25:
  analogWrite(PWM,255);        //cotxe aturat
}

if (atencio >=25){              //si l'atenció és igual o superior a 25...
  if (atencio<=75){            //i inferior o igual a 75 (per tant, de 25 a 75):

invertida=100-atencio;          //invertim la dada atenció

voltatge= map(invertida, 25, 75, 0, 255); //Mapegem* l'atenció invertida

  analogWrite(PWM,voltatge);    //Ofereim el voltatge resultant del mapejat de l'atenció invertida
  }
}

if (atencio >75){               //si l'atenció és superior a 75
  analogWrite(PWM,0);           //velocitat màxima
}

                                //paralel·lament, amb la dada atencio obtinguda, engegarem els LEDs...
                                //...si compleixen les condicions adients

if (atencio < 10){              //si l'atenció és inferior a 10 (per tant, de 0 a 9)
  digitalWrite(1,HIGH);         //LED 1 = encès
}

```

**PART NOVA**

77 Arduino/Genuino Uno on COM6

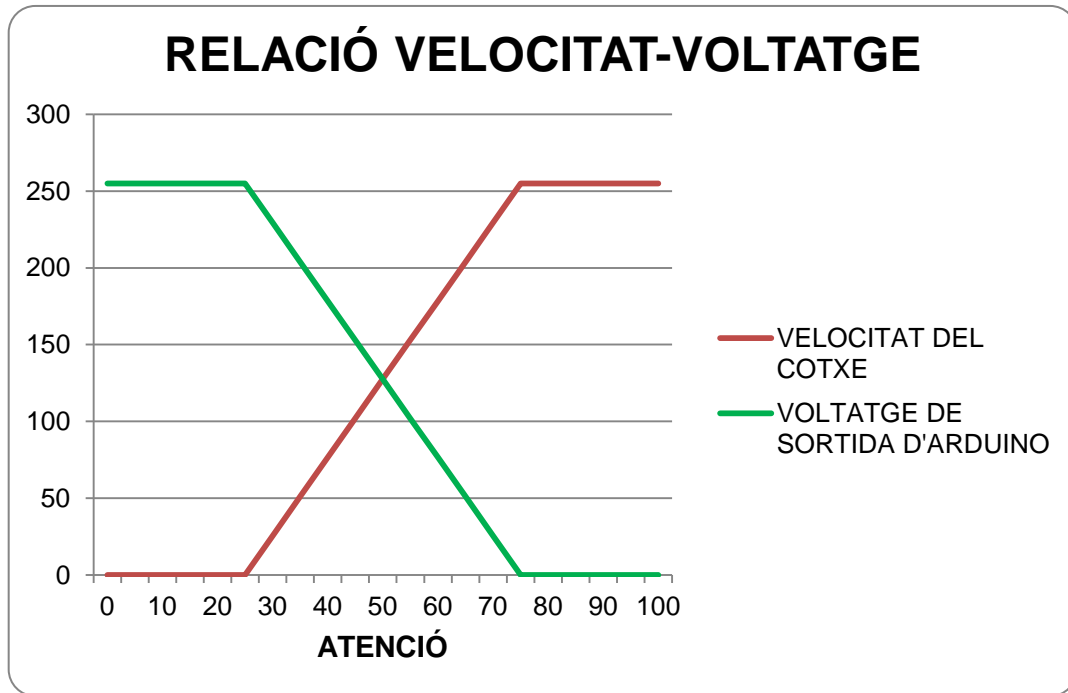
Aquesta part requereix certa explicació:

En primer lloc, hem de tenir en compte que, en la placa electrònica, l'*aixeta* que regula el voltatge del Scalextric funciona inversament. Com més voltatge oferim des d'Arduino, menys n'hi arriba als rails. És per això que quan l'atenció és baixa, oferim el màxim de voltatge (255) des d'Arduino, i quan aquesta és alta oferim un voltatge de 0 a la placa electrònica, que significa que arriba el voltatge màxim al circuit. (l'explicació continua a la pàgina següent)

Les 3 instruccions de la segona condició (atenció entre 25 i 75) també són explicades per aquest fenomen:

En primer lloc, fem la inversa de l'atenció (atenció=0  $\rightarrow$  inversa=100, 27 $\rightarrow$ 73, 63 $\rightarrow$ 37...). Posteriorment *mapegem* (extrapolem) la inversa de 25-75 a 0-255 i oferim el voltatge a la placa electrònica.

Per veure-ho més gràficament, podem fixar-nos en aquest gràfic:



## 5.2 COMPONENTS DEL SCALEXTRIC INMIND

### 5.2.1 NEUROSKY MINDWAVE HEADSET

El NeuroSky Mindwave Headset, del qual ja n'hem vist imatges durant el desenvolupament del treball, és l'element que s'encarrega de captar les ones elèctriques del cervell.

Les característiques i especificacions del Headset es troben, en anglès, en el següent link (<http://store.neurosky.com/products/mindwave-1>) o en el codi QR i, traduïdes al català, són les següents:

#### **Característiques:**

- Lleuger
- Inalàmbic (sense fils)
- Biosensors passius i segurs
- 8 hores de vida de pila AAA
- Inclou un CD amb 10 *apps* sobre neurociència.
- Sistemes operatius suportats:

Windows: XP, Vista, Windows 7, Windows 8, Windows 8.1

Mac: 10.7.5, 10.8.x, 10.9.x, 10.10, 10.11



#### **Descripció general del *hardware* o maquinari:**

- Auricular portàtil captador d'ones cerebrals (EEG)
- Mòdul TGAM1 amb TGAT1 ASIC
- Aparellador automàtic sense fils amb l'ordinador
- Identificació del *casc* estàtica
- Pila AAA
- 6-8 hores de vida útil de la pila

### **Software inclòs:**

- Meditation Journal
- SpeedMath
- BlinkZone
- Schulte
- SpadeA
- Find Number
- MindHunter
- Man.Up
- MindtyAnt
- Jack's Adventure

### **I gratis online:** (contingut descargable)

- Developer Tools
- Visualizer 2.0
- NeuroBoy
- MyndPlay

Nota: El Mindwave està destinat a ser usat en zones de 60 Hz, com els EEUU. Si es vol utilitzar el Mindwave en zones de 50 Hz, com Europa, sis plau adquireixi el seu Mindwave des de la nostra botiga europea a <http://www.mindtecstore.com/index.php/en/shop/neurosky>

### **Especificacions:**

- Pes: 90 g
- Mesures del sensor: Alçada: 225 mm; Amplada: 155 mm; Profunditat: 92-165 mm
- Potència mitjana: 30 mW; Potència màxima: 50 mW
- Freqüència RF 2.420-2.471 GHz

- Potència màxima RF: 6dBm
- Velocitat de transmissió de dades RF: 250 kbit/s
- 5% de pèrdues de paquets de bytes en la connexió inalàmbrica
- Velocitat de transmissió en bauds: 57.600 bauds
- Rang d'entrada màxima de senyal EEG 1mV pk-pk
- Rang de filtres de hardware: 3 Hz - 100 Hz
- Resolució ADC: 12 bits
- Freqüència de mostreig: 512 Hz
- Taxa de càlcul eSense: 1 Hz

### **Mesures:**

- Senyal *cru* (sense modificacions)
- Espectre d'ones definides per la neurociència (Alpha, Beta, etc.)
- Mesurador eSense per l'Atenció
- Mesurador eSense per la Meditació
- Detecció eSense dels parpelleigs
- Detecció de connexió

### **Preu:**

\$79.99

## **5.2.2 ENTORN DE PROGRAMACIÓ: PROCESSING**

Un dels dos entorns de programació utilitzats en la realització d'aquest treball és el de Processing, un llenguatge de programació similar a C que se sol utilitzar per veure representacions gràfiques en la pantalla. En aquest projecte,

com ja s'ha dit en el desenvolupament, s'utilitza per poder tractar les dades que capta el *NeuroSky Mindwave Headset*.

Aquest entorn de programació està disponible per diversos sistemes operatius, tals com Windows (a 32 o 64 bits), Linux (a 32 bits, 64 bits o ARMv6hf) o Mac OS X. El que s'ha utilitzat el Processing destinat a Windows 32 bits, concretament la versió *Processing 3.0.1*, el link de descàrrega de la qual és el següent:

<http://download.processing.org/processing-3.0.1-windows32.zip>

### 5.2.3 ENTORN DE PROGRAMACIÓ: ARDUINO

L'altre entorn de programació emprat en aquest treball és, com ja s'ha insistit en el desenvolupament del treball, l'entorn Arduino. Com que ja s'ha explicat suficientment al llarg d'aquest treball, em limitaré a dir que la versió utilitzada ha estat la de *Arduino 1.6.7* per a Windows, el link de descàrrega de la qual és el següent:

[https://www.arduino.cc/download\\_handler.php?f=/arduino-1.6.7-windows.exe](https://www.arduino.cc/download_handler.php?f=/arduino-1.6.7-windows.exe)

Arduino també està disponible per Mac OS X (10.7 o superior) o Linux (32 i 64 bits).

Dit això, també és important nomenar els components, el *hardware*, de l'àmbit Arduino utilitzats en la realització d'aquest treball. Són els següents:

- Placa *Funduino UNO*
- 10 LEDs vermells
- 10 resistències de 110  $\Omega$
- Protoboard*
- Cablejat

## 5.2.4 PLACA ELECTRÒNICA ARDUINO → SCALEXTRIC

El funcionament de la **placa electrònica** que fa de pont entre la placa Arduino i el Scalextric està explicat al subapartat 2.5.3 Arduino → Scalextric. Tot i així, aquest apartat de l'annex pot servir per explicar com s'ha construït la placa.

En primer lloc, es va haver de trobar el diagrama definitiu construir, que tal com s'indica en el subapartat esmentat anterior és aquest:

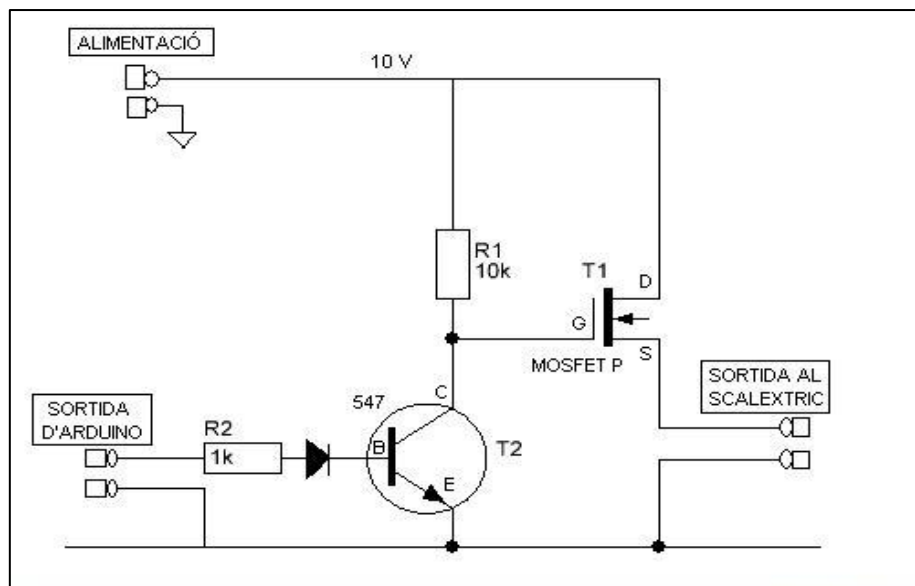


Diagrama de la placa electrònica definitiva *[imatge pròpia]*

Els components d'aquesta placa electrònica són:

- 3 connectors blaus del tipus Screw Terminal Block de 3 pins. El pin del mig no té ús en aquesta placa
- 1 resistència de 10k  $\Omega$
- 1 resistència de 1k  $\Omega$
- 1 transistor MOSFET P IRF4ZNN (Datasheet: <http://dom.cat/r0b> )
- 1 transistor BC547 (Datasheet: <http://dom.cat/r40> )
- Cablejat negre i vermell



Un cop aconseguits els components, s'ha de procedir a connectar-los entre si. Per fer-ho, tots aquests components han estat soldats adequadament en una placa de soldadura lliure, és a dir, sense circuit integrat. Això significa que les connexions entre els components s'havien de fer mitjançant cablejat per sota de la placa, tal com indica alguna imatge present en el treball.

Totes les soldadures s'han fet amb un soldador d'estany i tot el kit que l'acompanya: el mateix estany, una estructura on poder reposar el soldador o una esponja per netejar el soldador.

El preu de la construcció d'aquesta placa ha estat pràcticament de 0, ja que molts dels components utilitzats ja es trobaven disponibles en kits d'Arduino de l'institut o es podien comprar per un preu bastant baix (menys de 10 € en total), i el kit de soldadura ja era propietat de l'institut.

### **5.2.5 SCALEXTRIC**

Per poder construir el prototip de Scalextric Inmind, òbviament fa falta una Scalextric. S'ha utilitzat un Scalextric que ja es tenia, no se n'ha comprat cap de nou perquè no era necessari, en qualsevol tipus de Slot s'hagués pogut realitzar el treball. Així doncs, els components utilitzats han sigut:

- Powerbase
- Transformador (inclòs al lot del Scalextric)
- Pista de Scalextric
- Cotxe de Scalextric

Tot i així, el model de Slot concret que s'ha emprat en aquest Treball de Recerca ha estat un Scalextric "Le Mans" dels anys 90 propietat del tutor d'aquest treball, tot i que es podria haver utilitzat qualsevol model.

## **6. AGRAÏMENTS**

Un cop acabat el treball, m'agradaria agrair l'esforç i les molèsties causades a totes aquelles persones que m'han ajudat en el desenvolupament d'aquest prototip de Scalextric Inmind:

*A la meva família i amics, que han estat sempre al meu costat tant si el treball anava bé com si no, animant-me sempre que sorgien dificultats.*

*Al Roger València per ajudar-me en les soldadures de la placa electrònica "Arduino →Scalextric".*

*A la gent anònima dels fòrums de Processing, Arduino i, sobretot, Automodelismo Slot, que han respost desinteressadament tots els dubtes que els hi he plantejat.*

*Al Jordi Virgili, per ajudar-me en el disseny de la placa electrònica i per l'assessorament als inicis del treball, concretament en l'escolliment del casc adient per realitzar el treball.*

*I, per últim, al tutor d'aquest Treball de Recerca, sense el qual hagués estat impossible realitzar aquest projecte amb èxit.*

