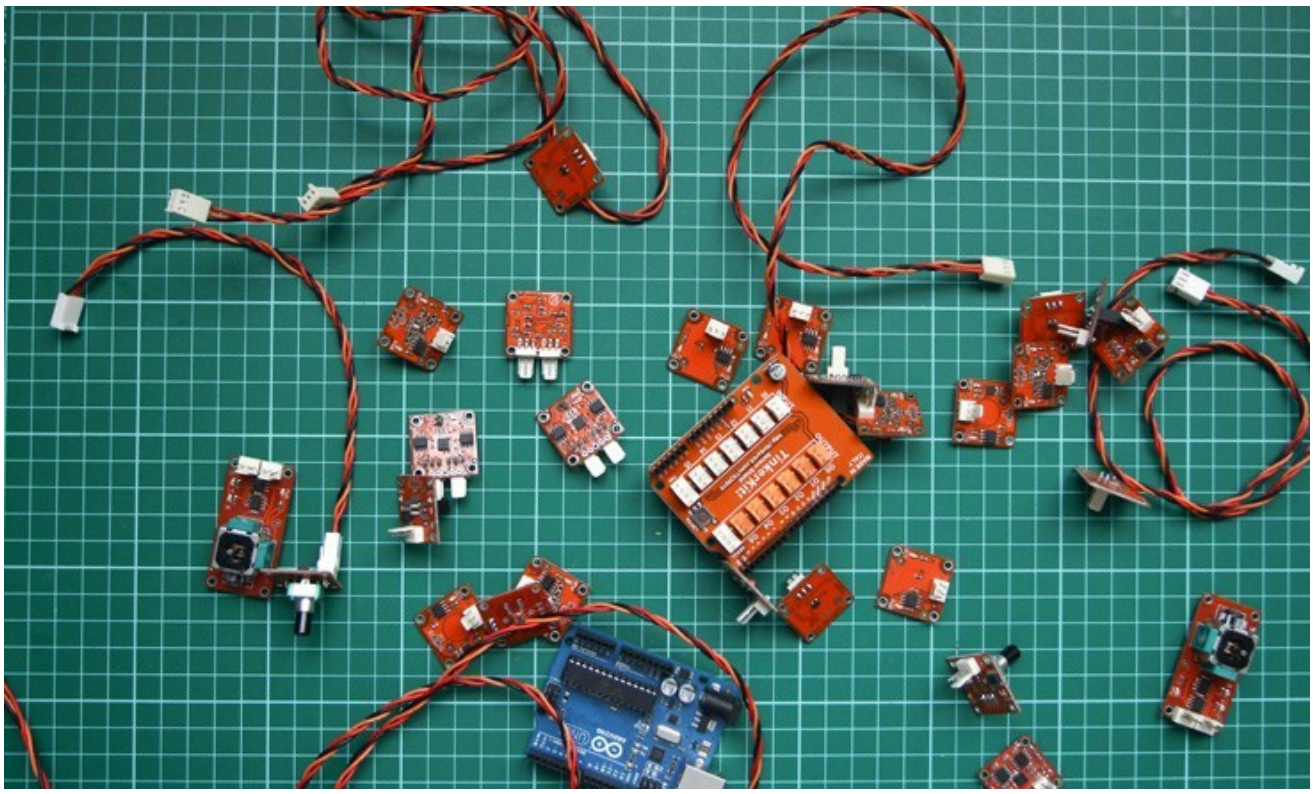
Guia d'introducció a la programació amb Arduino TinkerKit

Aquesta guia ha estat elaborada a partir de la traducció dels tutorials elaborats per Canada Robotix i oferts en el seu blog.

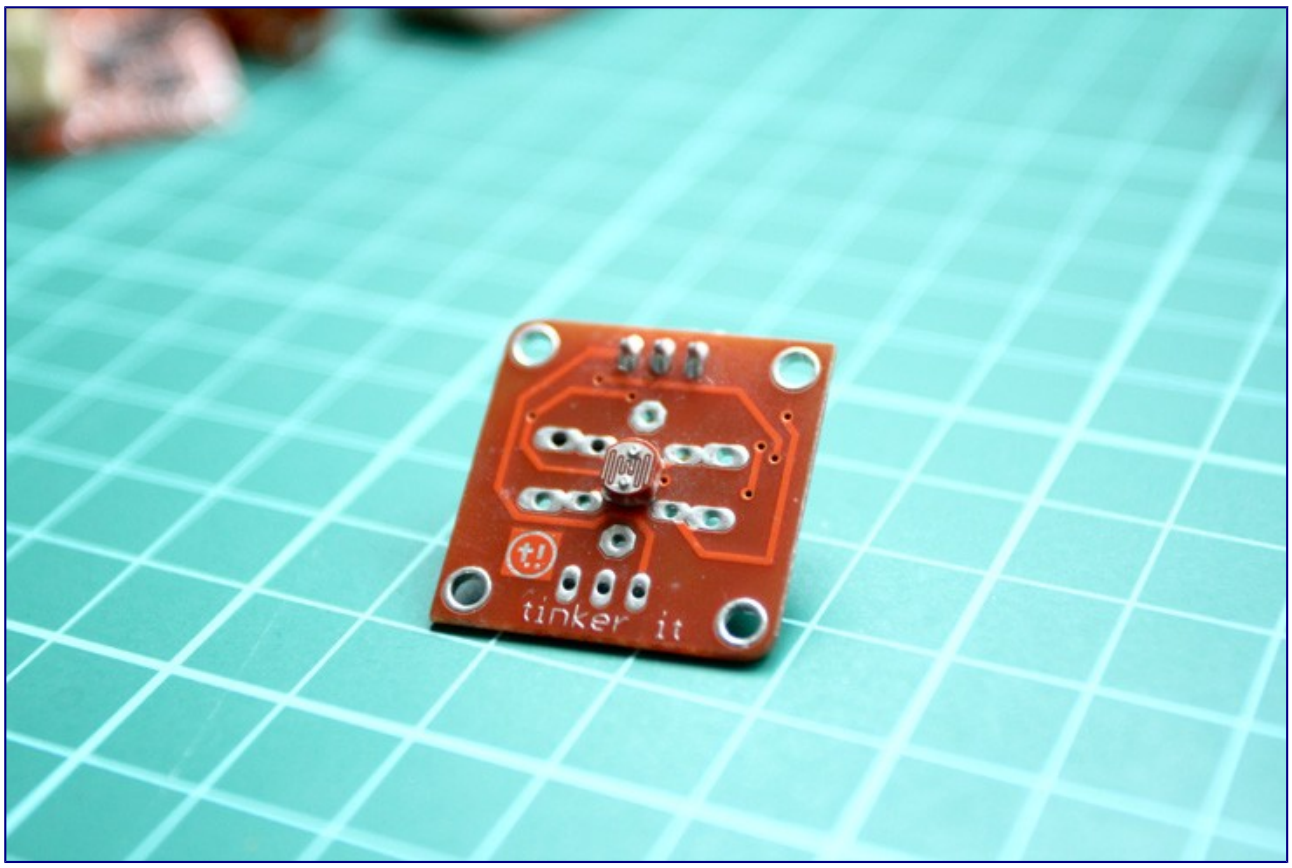
Índex

Introducció: TinkerKit i Arduino

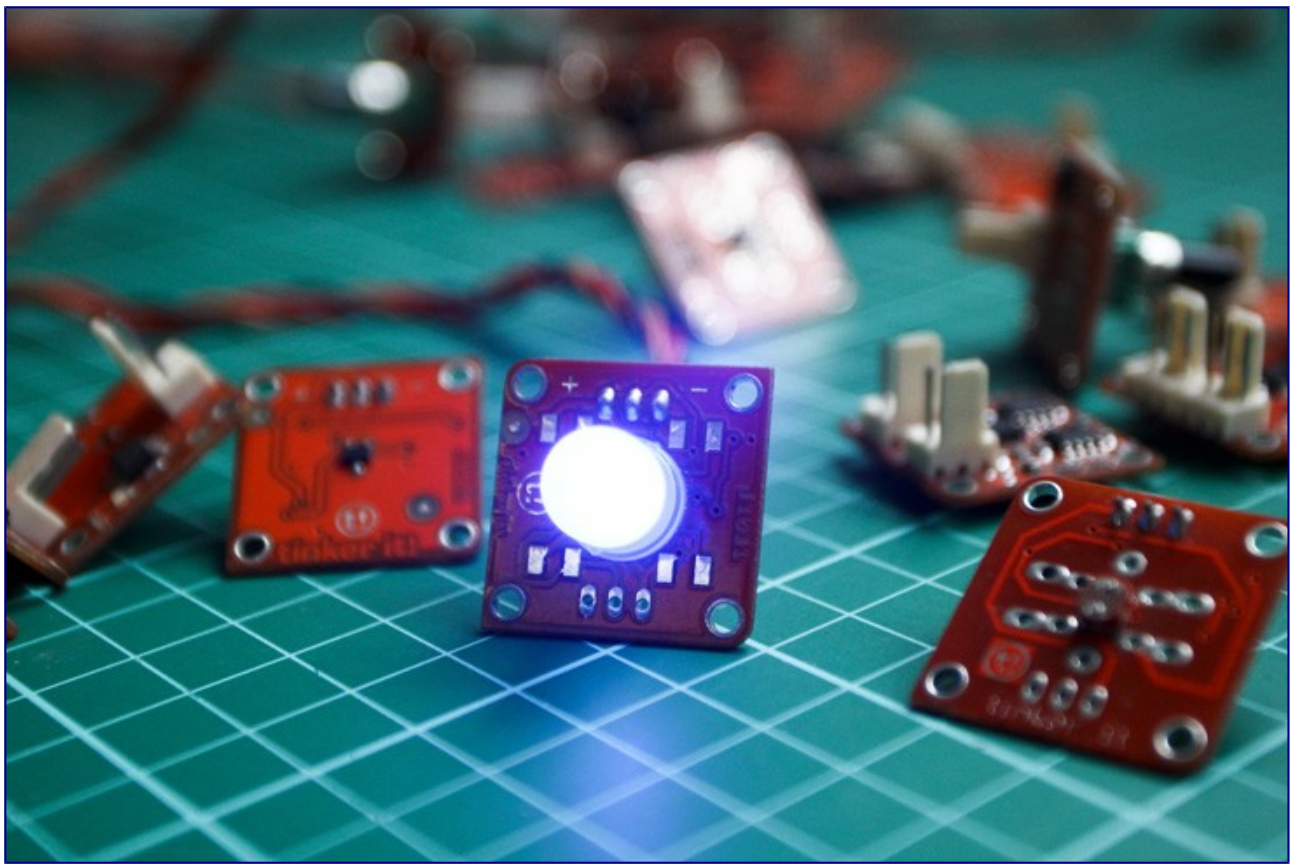


TinkerKit és una eina usada per construir productes interactius d'una manera simplificada en molt poc temps. Funciona amb una placa d'Arduino amb la diferència que no has de construir físicament els circuits a partir de components de baix nivell com resistències o breadboards.. Gràcies al sistema modular de blocs només cal que els connectis fent servir els cables TinkerKit.

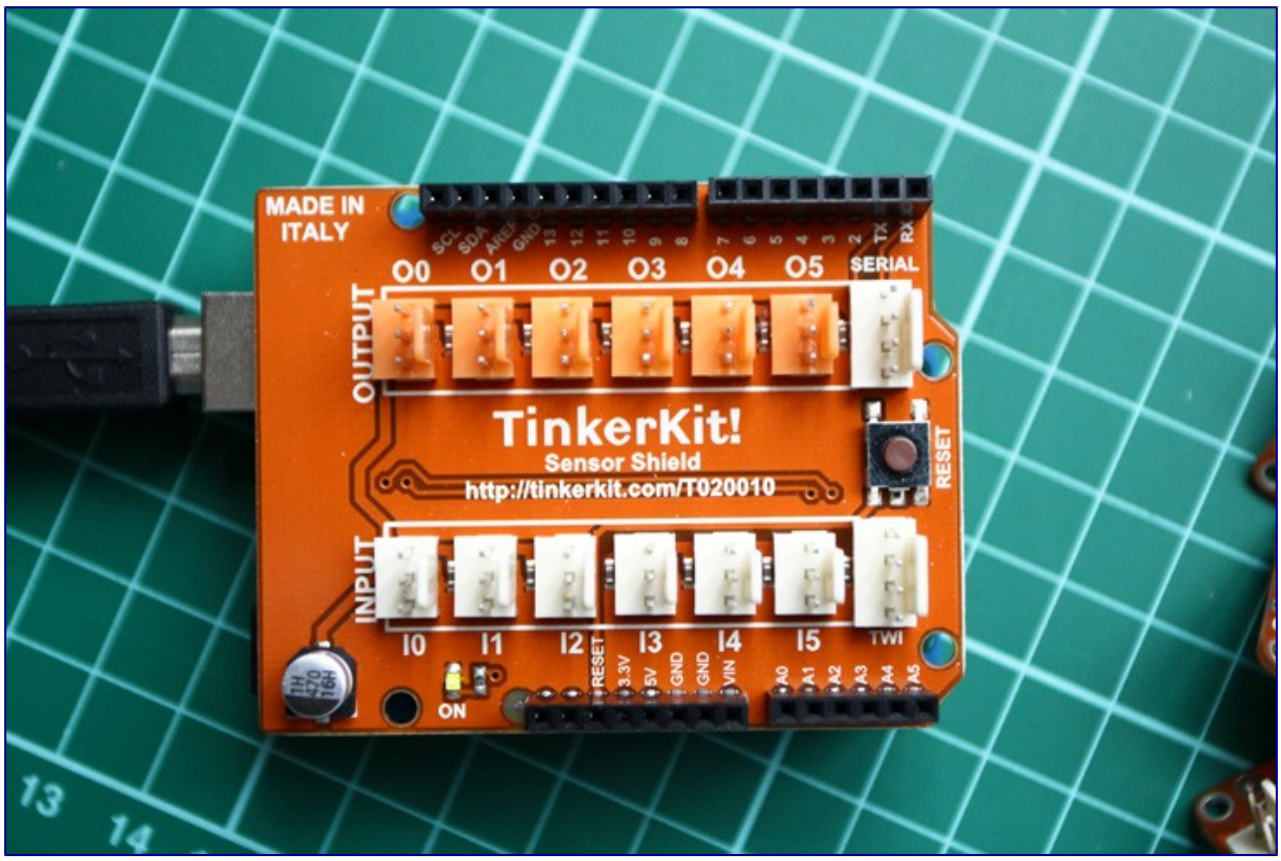
Els mòduls es divideixen en **sensor** i **actuadors**.



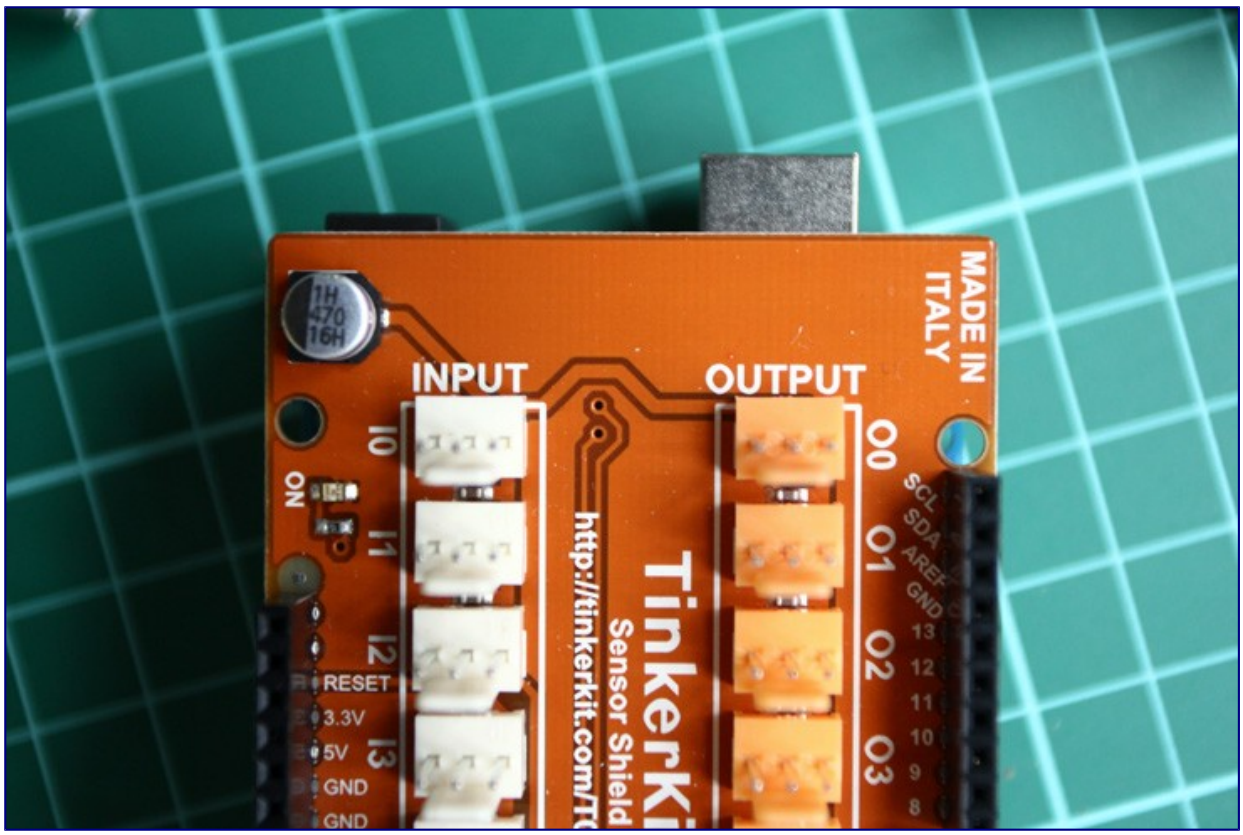
Els **Sensors** poden "sentir" els canvis en el seu entorn i els poden quantificar. Hi ha una gran quantitat de sensors, cada un amb les seves característiques. Poden percebre la llum (com el de la fotografia), la temperatura, el moviment, la pressió... A partir d'ara els anomenarem **INPUTS**.



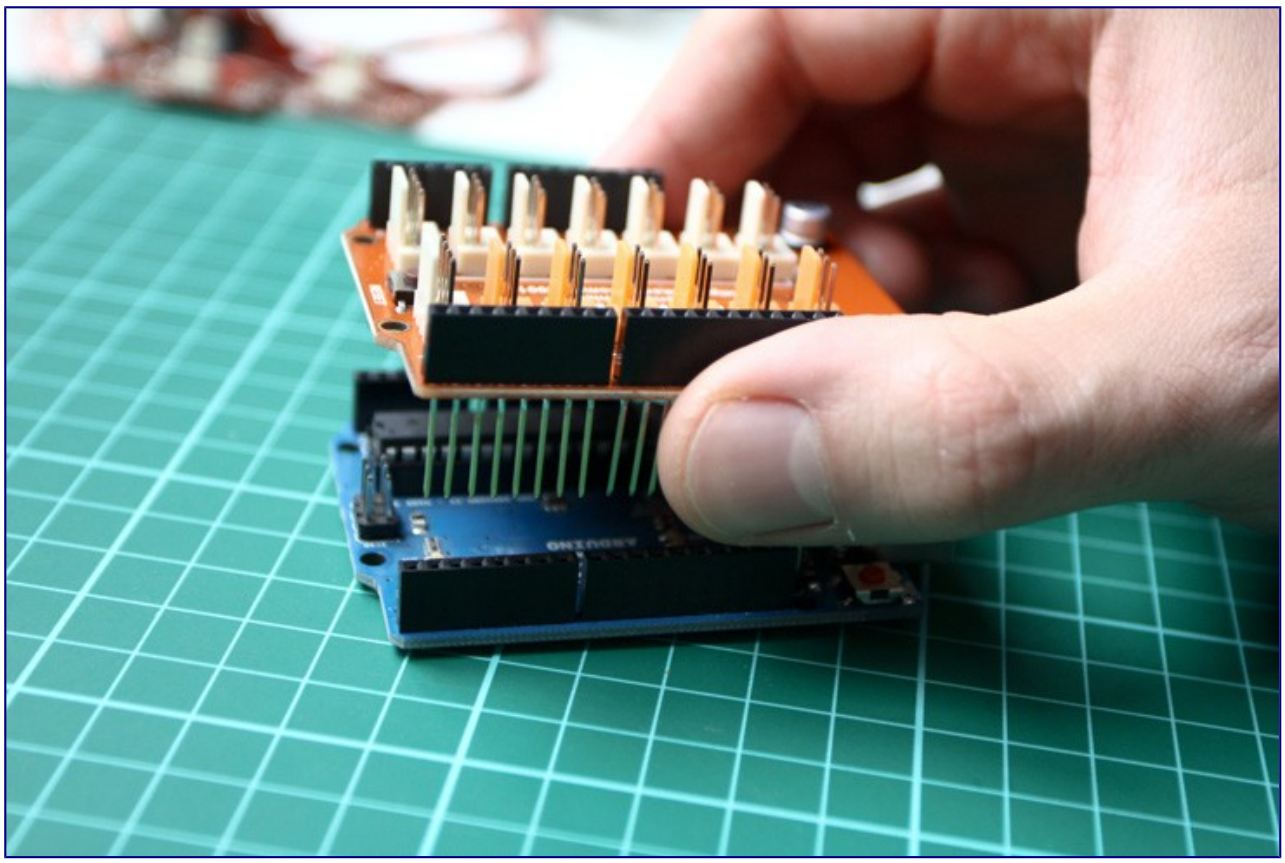
Els **Actuadors,** com ens diu el seu nom, fan una acció. Poden fer-se servir per diferents activitats, segons el tipus d'actuadors, des de fer llums a moviment i controlar altres dispositius. Els LEDs com els de la fotografia son actuadors, i els anomenarem a partir d'ara **OUTPUTS**.

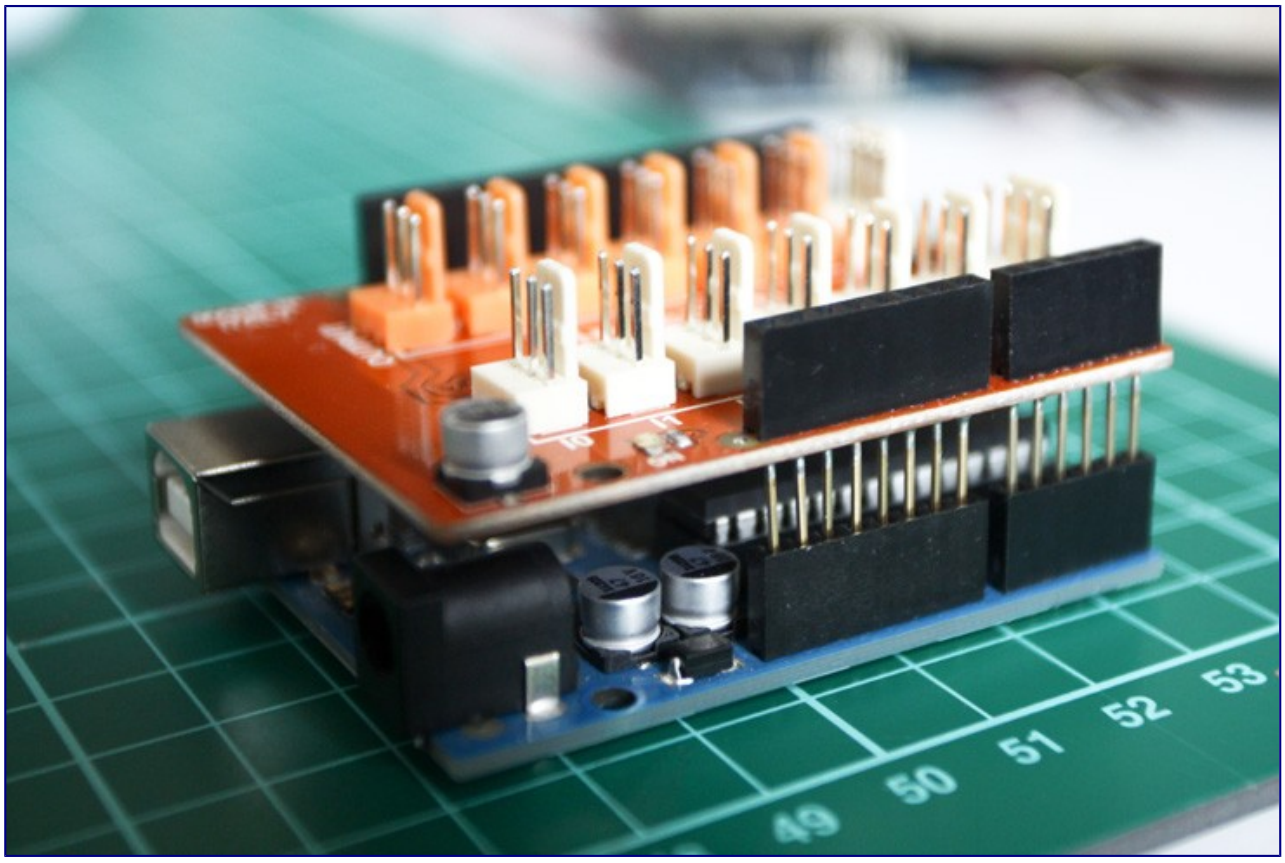


Aquesta és una **Shield de TinkerKit**, on s'endollen tots els components. De la mateixa manera que els mòduls, es divideixen en dues parts: per una banda tenim els INPUTS, i per l'altra els OUTPUTS. Cada part té sis ports, enumerats del 0 al 5. Els ports d'entrada (**INPUT**) són el **I0, I1, I2, I3, I4** i el **I5**; i els ports de sortida (**OUTPUT**) son el **O0, O1, O2, O3, O4** i el **O5**.



La Shield de TinkerKit és una extensió d'Arduino. Les Shields estan pensades per encaixar-les sobre de la placa i cada una té característiques i funcions diferents. Cal anar en compte no doblegar els pins quan encaixem la shield a sobre de la placa.



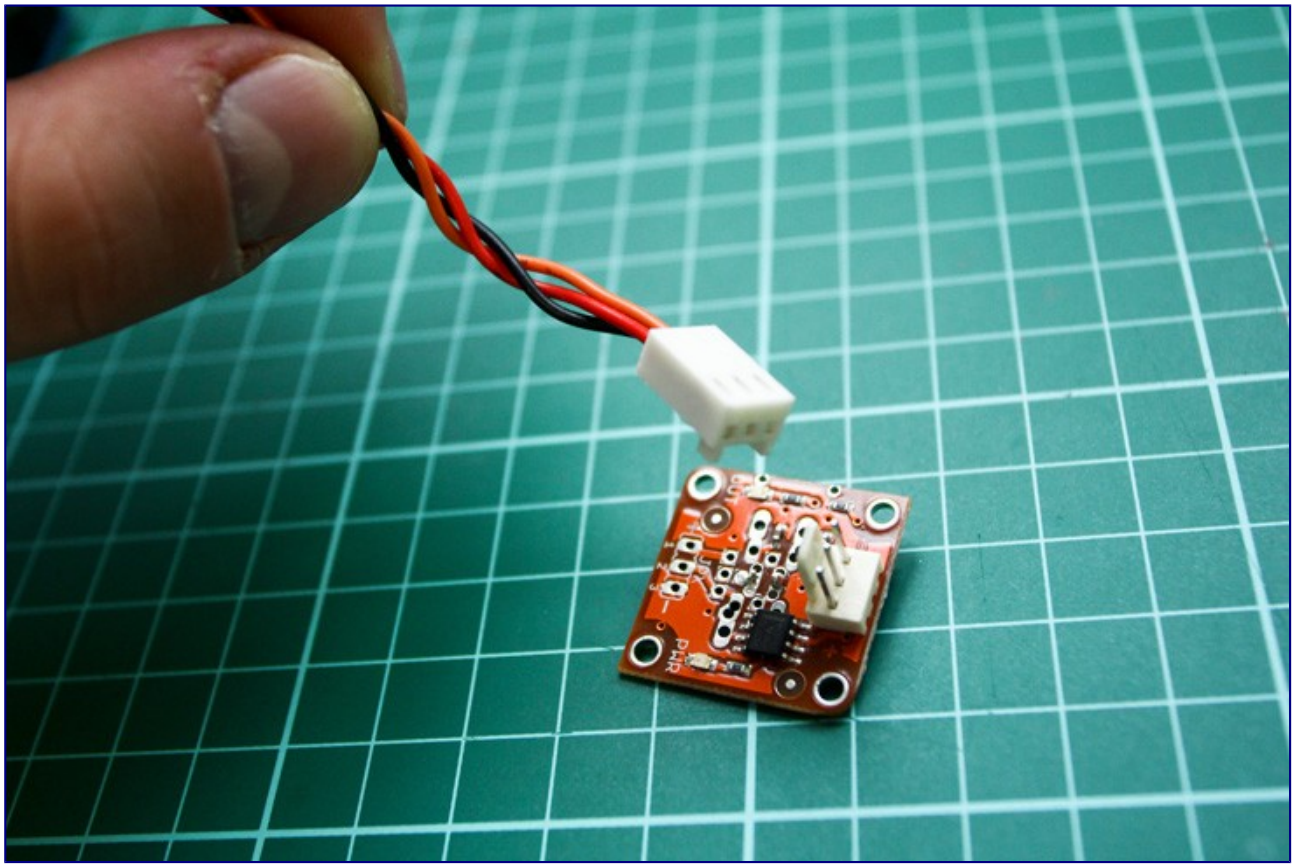

49
50
51
52
53

Hola Mon!

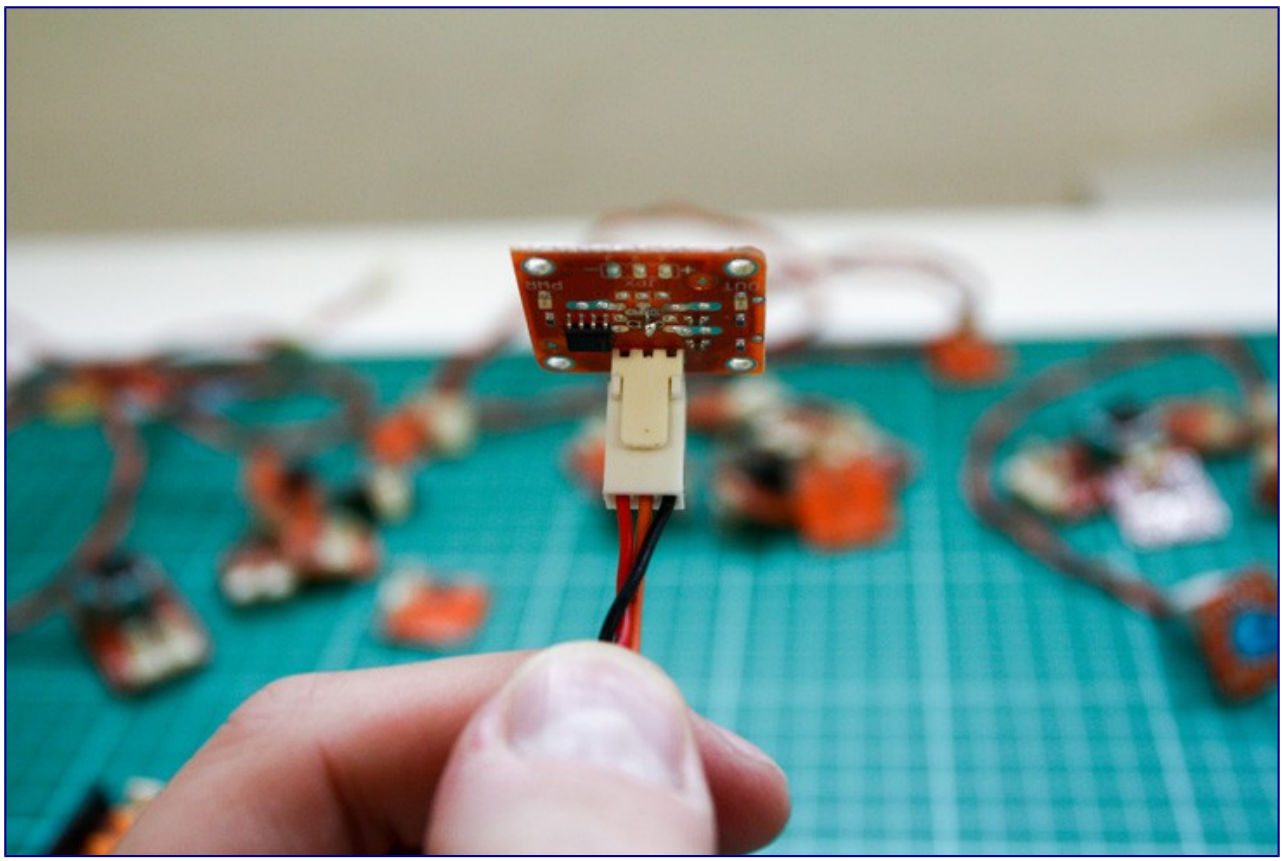
Ok, ara que ja ens hem avorrit amb tanta teoria, anem a fer alguna cosa. No entrarem en masses detalls sobre codi per que estem fent les primeres passes, però aprendrem com obrir un dels exemples i veurem alguns mòduls en acció.

Per aquest primer projecte necessitarem un LED (qualsevol led funcionarà, en les fotografies veiem un LED de 10mm blau) i un sensor de llum. Aquests són els que hem vist abans com a exemples de sensors i actuadors.

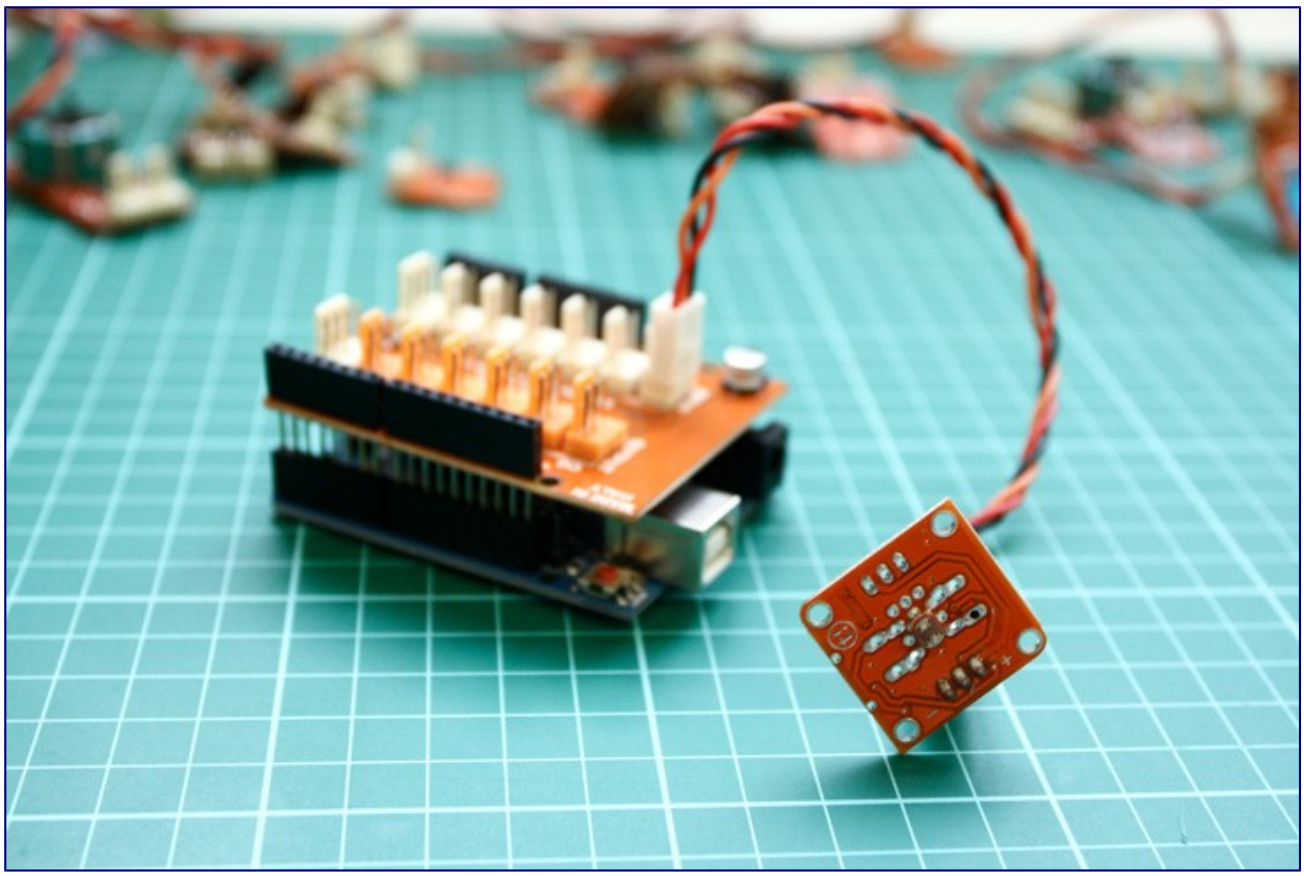
La primera cosa que hem de fer és encaixar -los fent servir un dels cables. Comencem amb el sensor de llum:



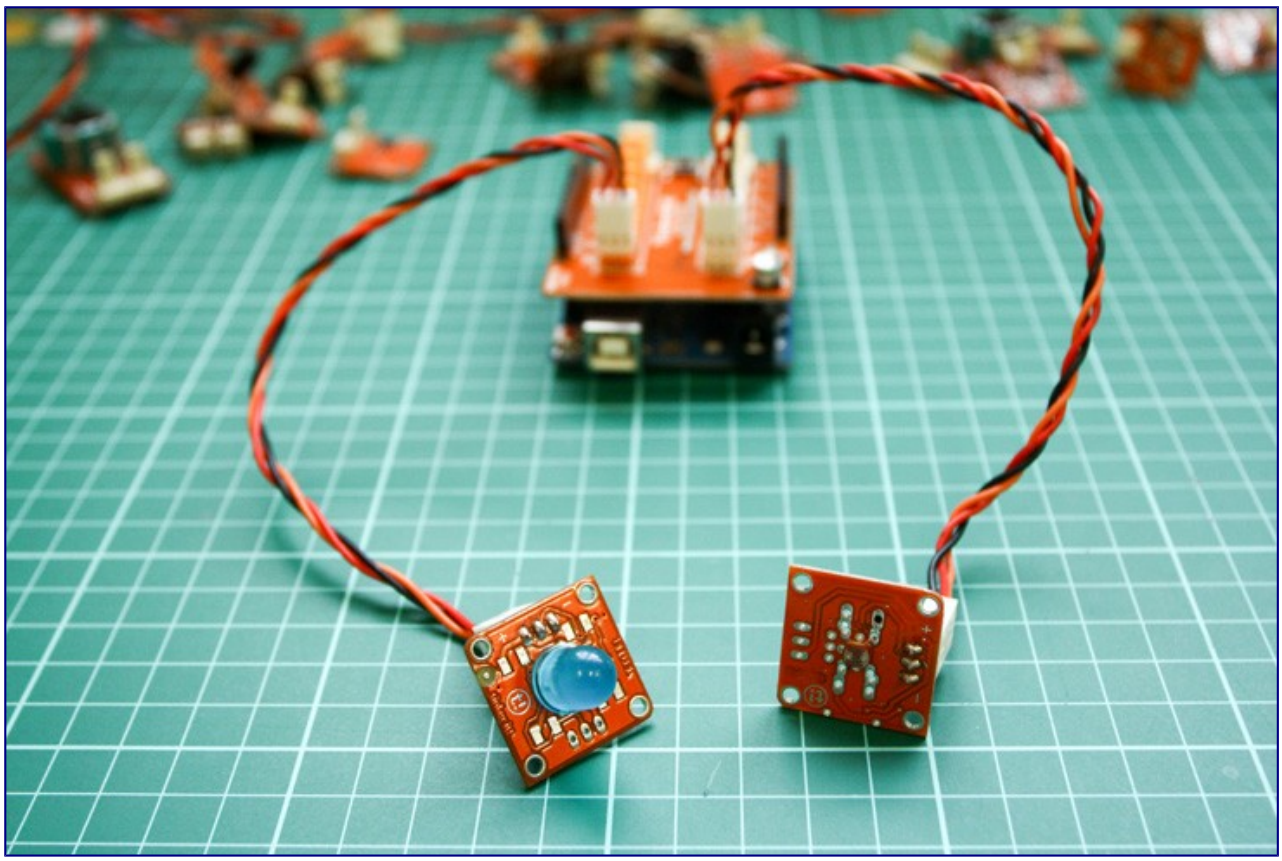
Assegura't que el cable està encaixat fins el fons del connector, sentiràs un "click". La connexió ha de ser ben fixe, com la següent fotografia.



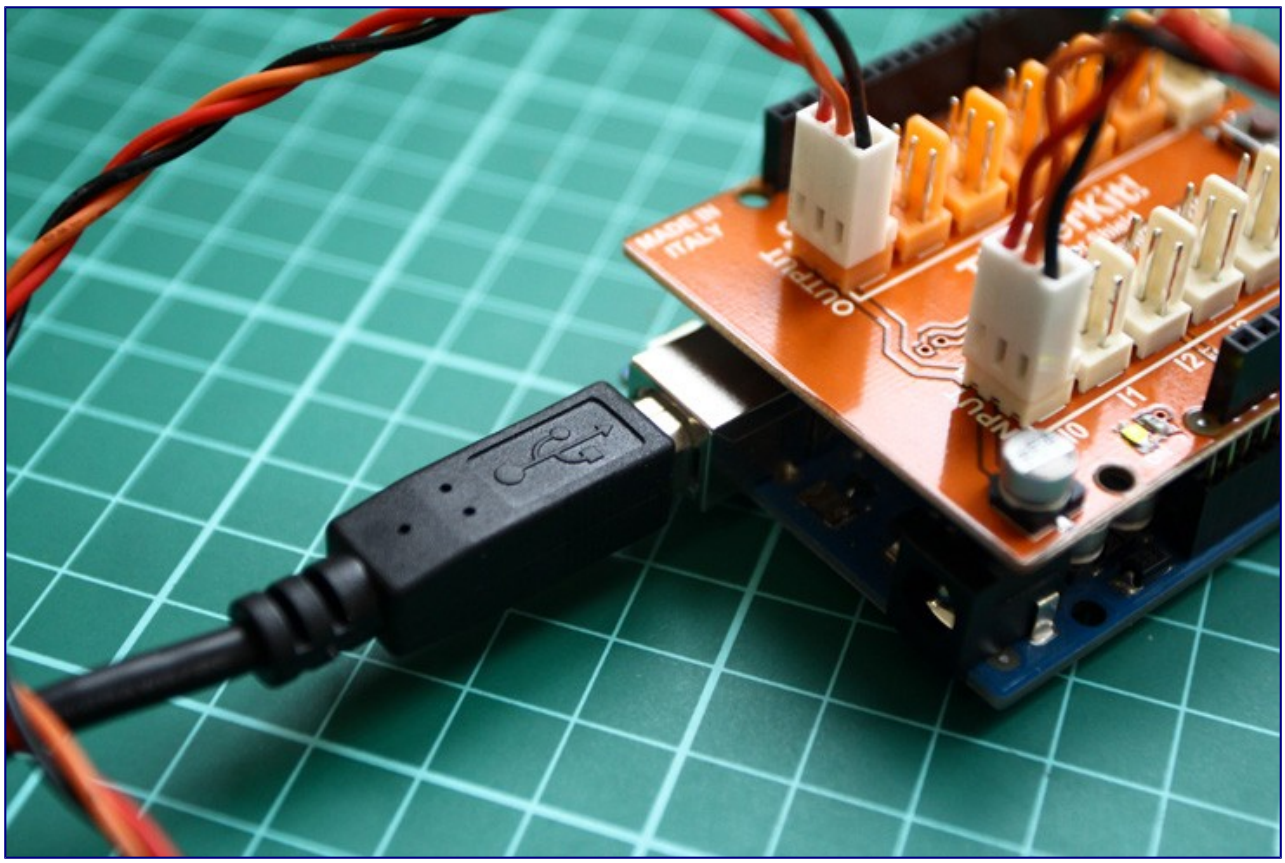
Ara connectem l'altre final del cable en el port **I0** de la **Shield de TinkerKit**. Com hem comentat abans, el sensor de llum ha d'anar connectat a un dels ports d'**INPUT**. En l'exemple que nosaltres seguim, el sensor ha d'estar connectat al port I0. Sempre és una bona pràctica començar per el primer port disponible, tot i que tots els ports poden funcionar.



Segueix les mateixes passes amb el LED, però enlloc de connectar-lo en el port I0, connecta'l en el O0. Tindràs algo similar a:

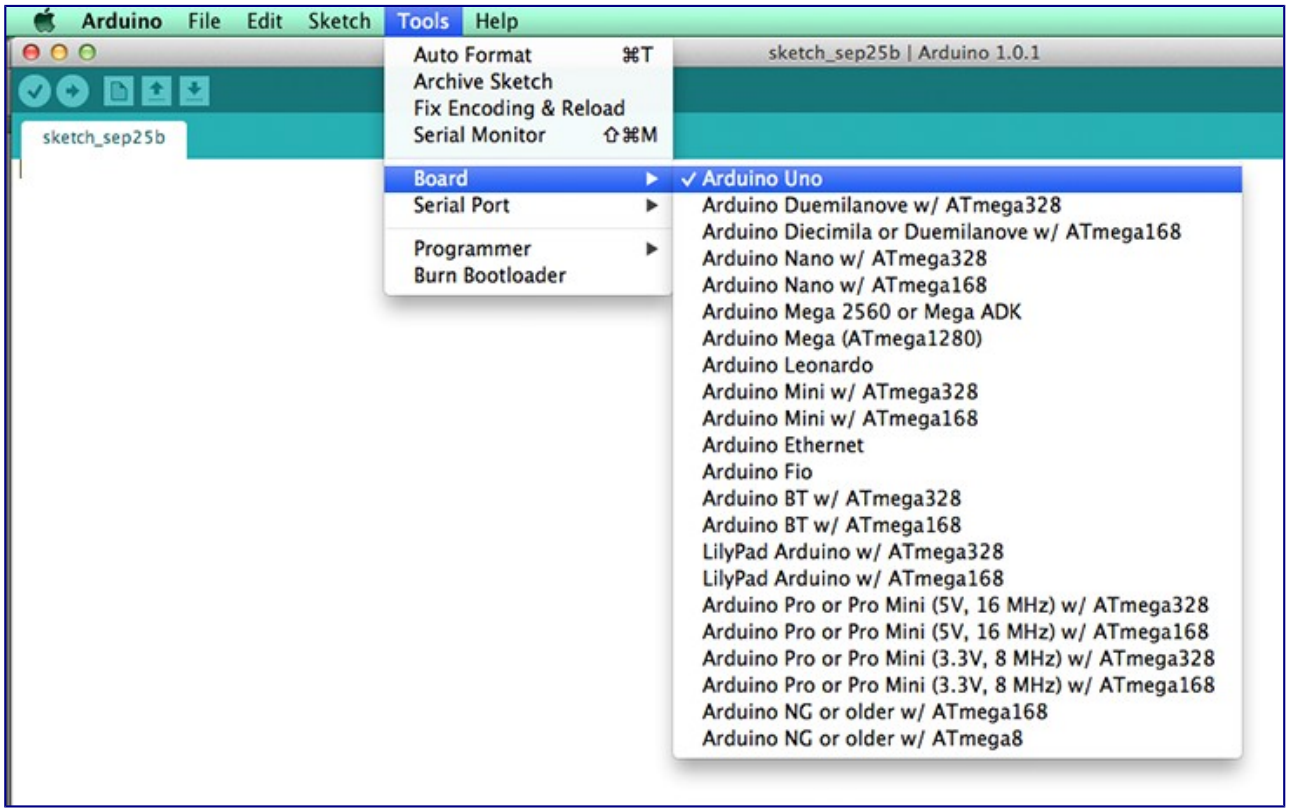


El que farem a continuació és fer que el LED reaccioni a la quantitat de llum que hi hagi a l'habitació. Per aconseguir això, li hem de donar als mòduls certes instruccions. La placa d'Arduino funciona com un petit cervell: quan li dones ordres, és molt eficient executant-les! Anem a connectar-la a un ordinador, fent servir el port USB com qualsevol dispositiu extern (un ratolí, una impressora...).

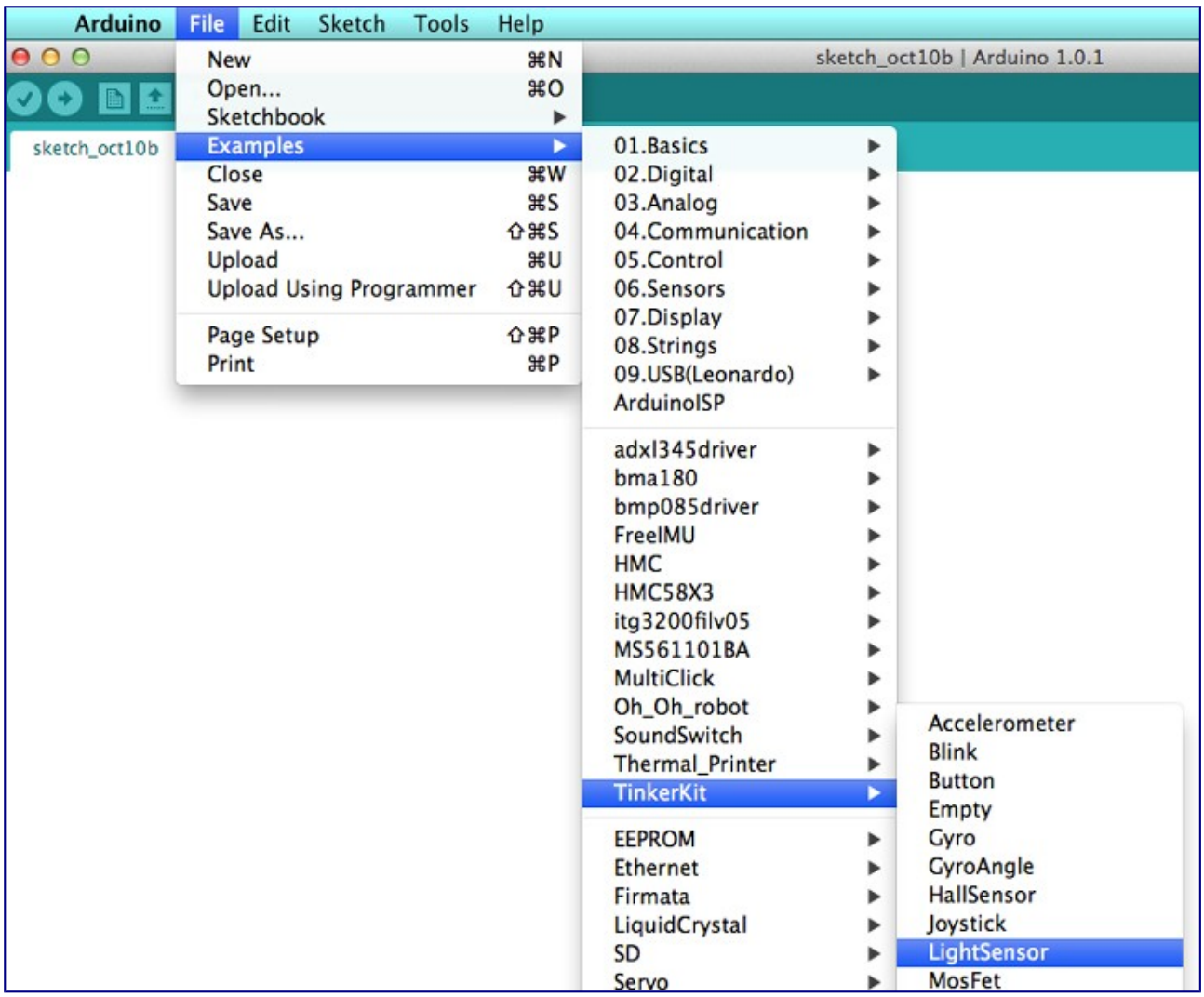


En l'ordinador, obre el programa **Arduino**, en el menú superior, desplega la opció **Eines > Placa** i selecciona la teva placa d'Arduino. En aquest tutorial estem fent servir una **Arduino**

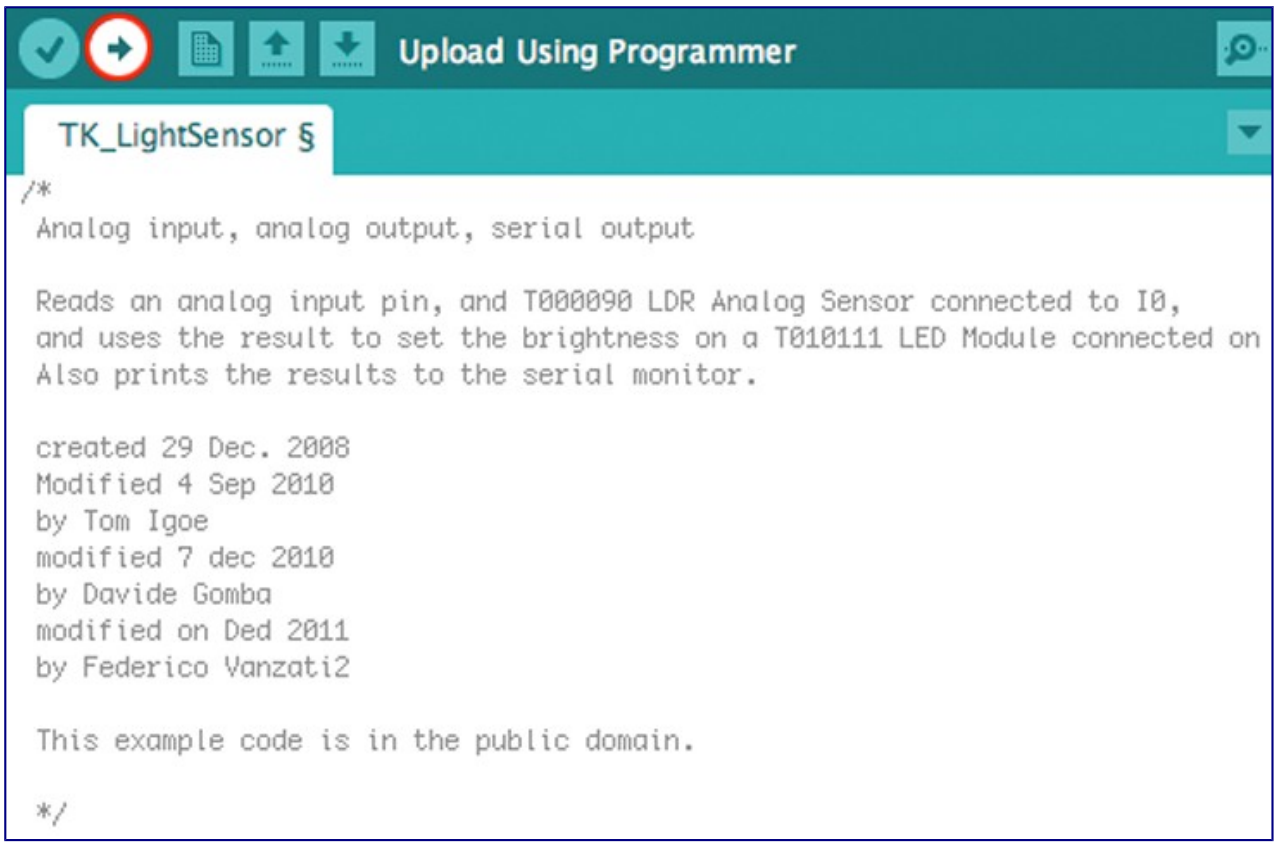
UNO.



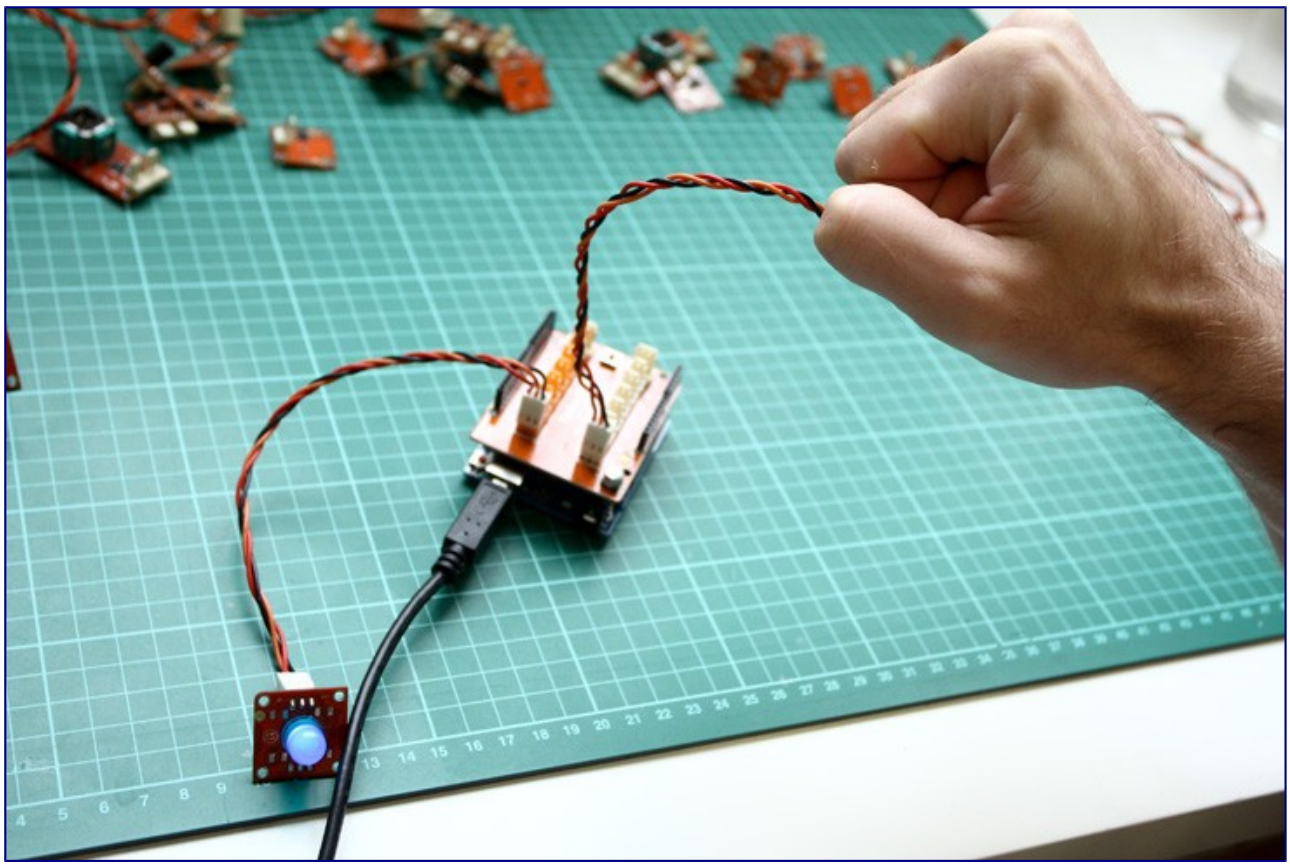
Casi ho tenim! Només hem de carregar l'exemple correcte en el programa. Clica a **Fitxer > Exemples >TinkerKit >LightSensor.** Si no veus el submenú TinkerKit és que no tens la llibreria correctament instal·lada.

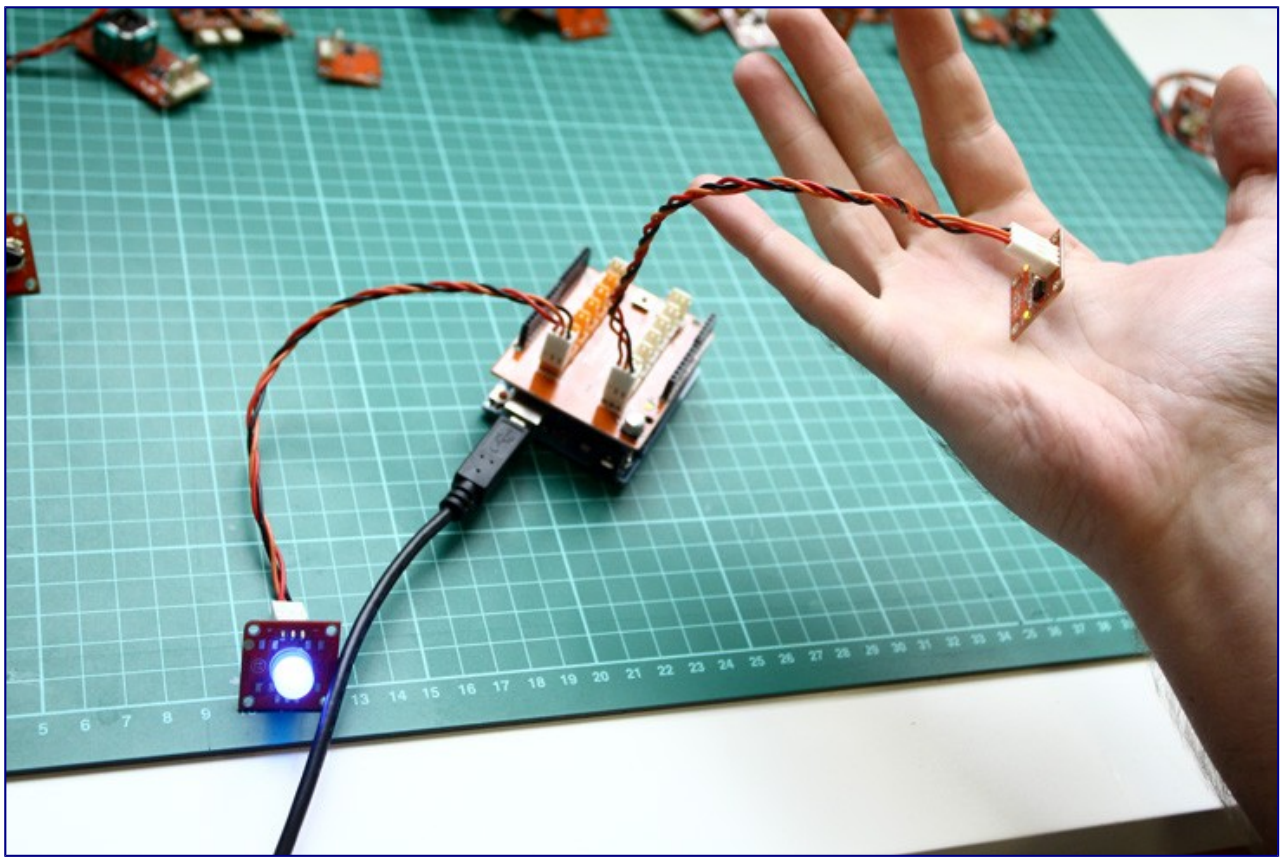


Veuràs com s'obre una nova finestra amb un fragment de codi, només cal carregar-lo a la placa fent servir el **Botó de càrrega** (ctrl+U)



Ben fet! Ara el teu LED ha de reaccionar a la quantitat de llum que rebi el sensor de llum. Prova de tancar-lo entre les teves mans, o apropa'l a una font de llum, si hi ha molta llum a l'habitació costarà veure com canvia.





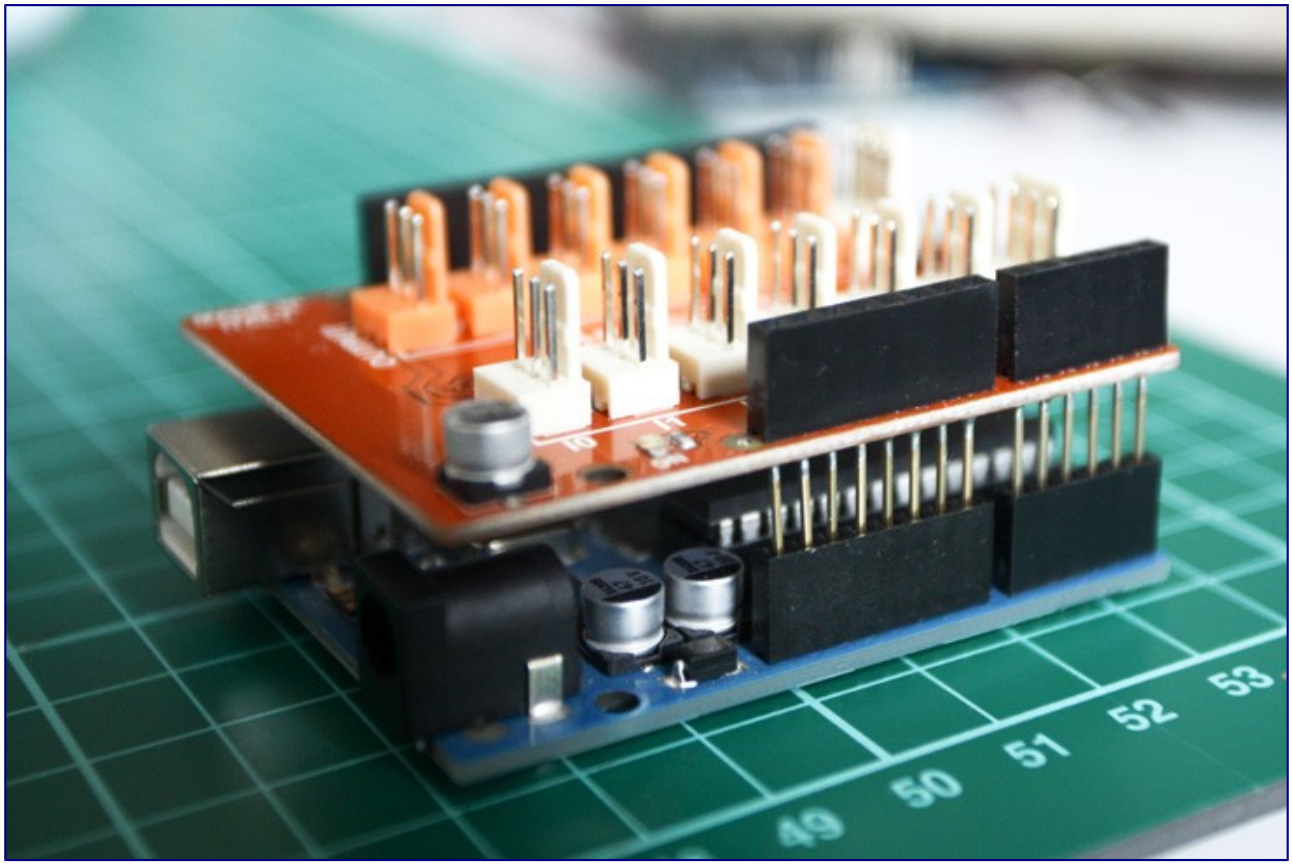
I ara?

Mira els altres exemples i explora la resta de mòduls. A cada un dels exemples, en les primeres línies de codi, hi ha indicacions sobre com connectar els mòduls. Explora el programa d'Arduino, juga amb els fragments de codi que tens modificant-los i veien què pot passar!

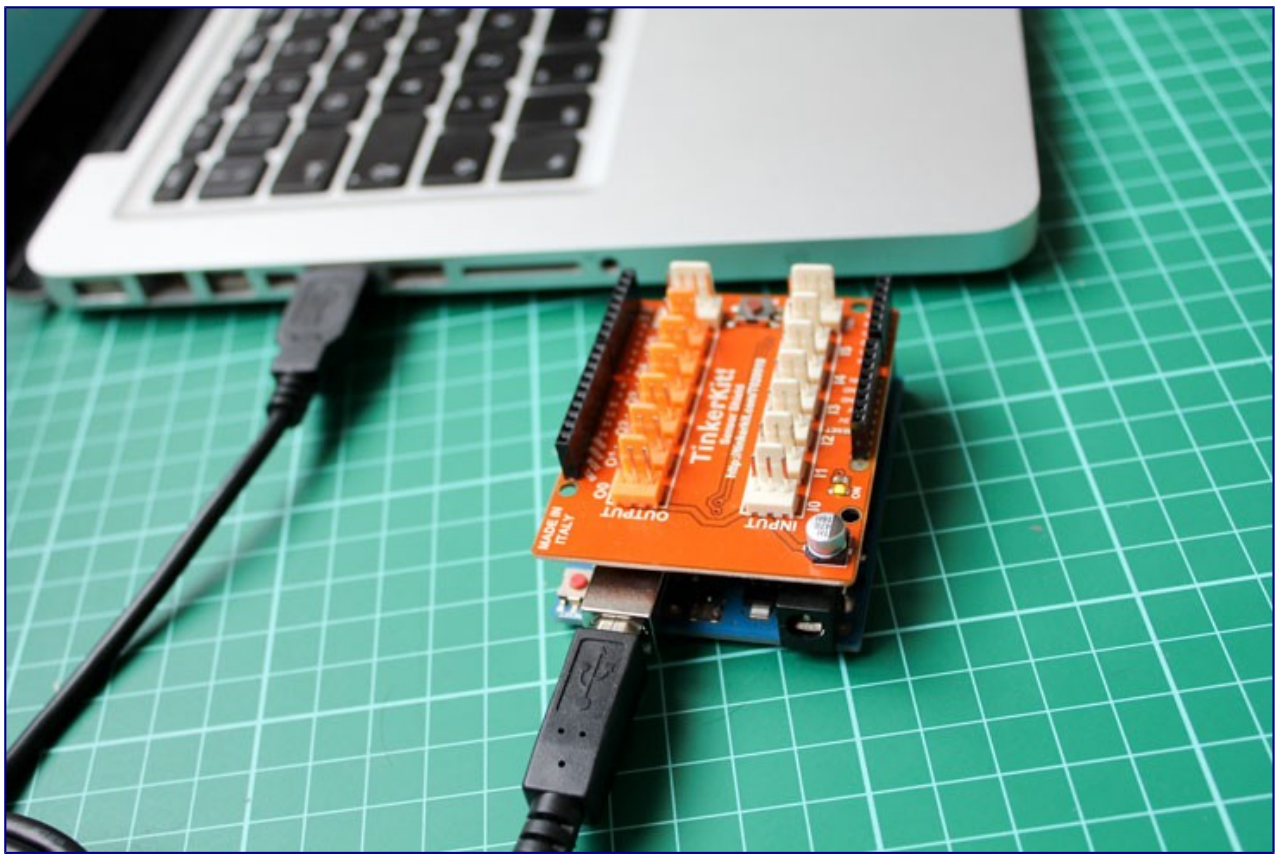
Comencem un projecte

Aquesta és una guia breu que t'explica **les tres passes** que has de fer abans de començar qualsevol projecte. Quan els hagis fet unes quantes vegades, ho faràs amb els ulls tancats, però sempre està bé tenir un bon recordatori.

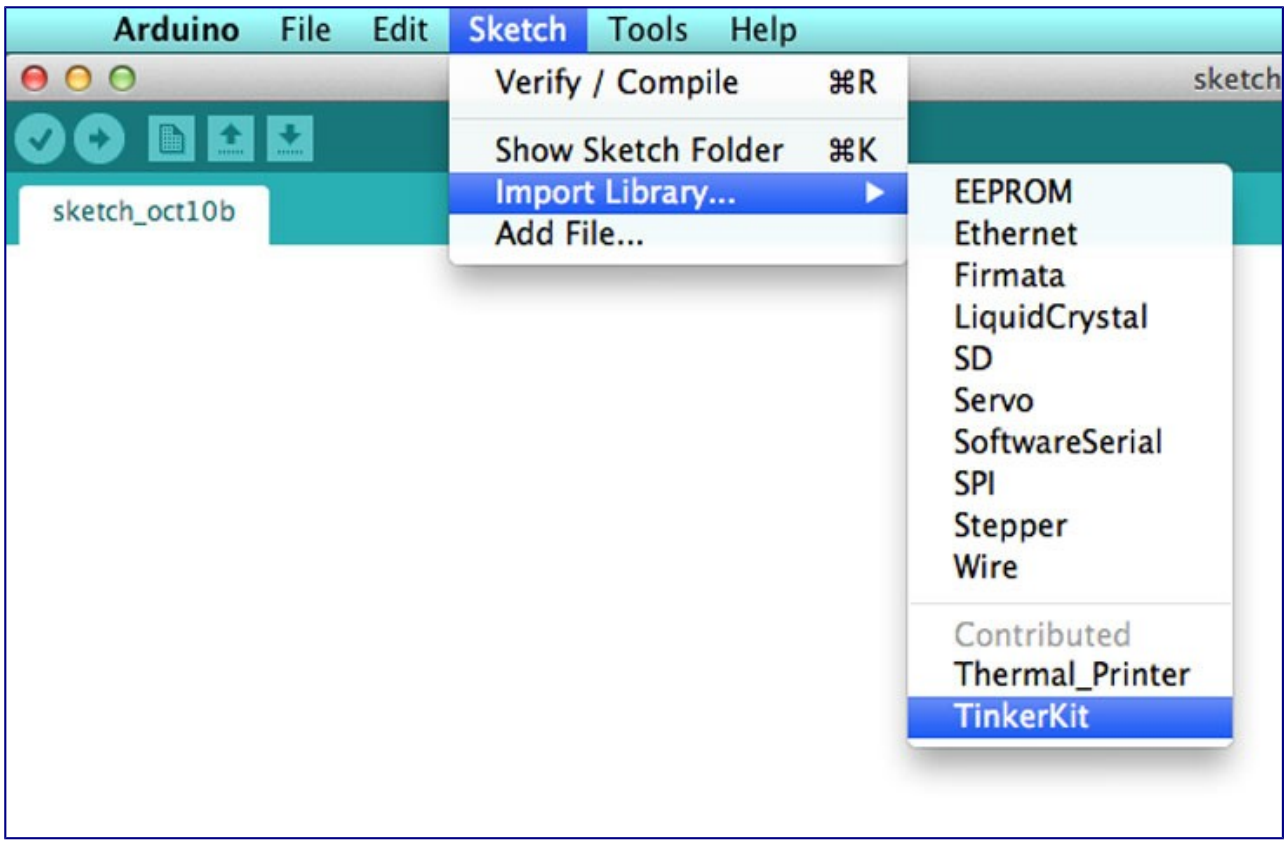
1. Encaixa la **shield** de Tinkerkit a la placa d'Arduino.



2. Endolla-ho al port **USB** del teu ordinador.

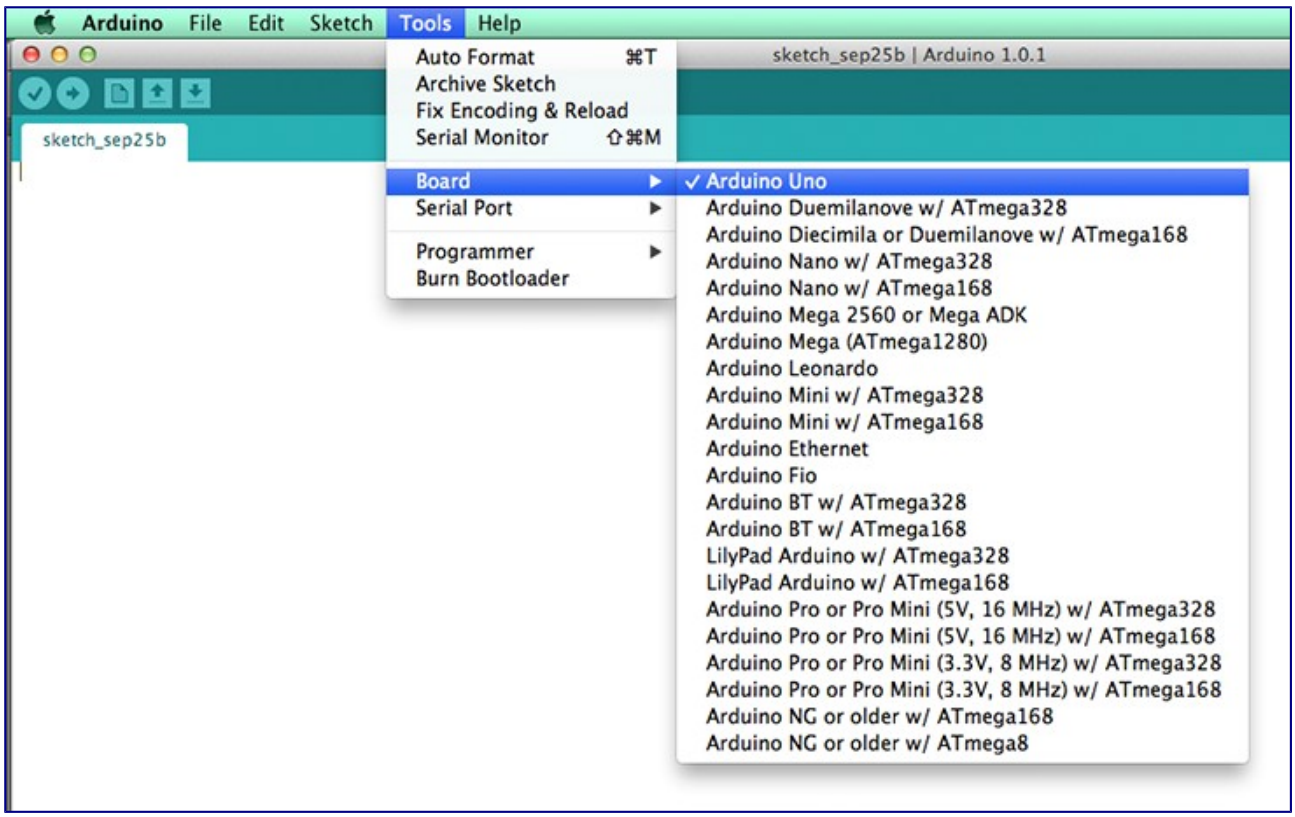


3. Obre el programa Arduino i importa la **llibreria** de TinkerKit



Quan la llibraria s'ha carregat correctament, veiem una línia de codi en el nostre editor:
#include <TinkerKit.h>

Si canvies la placa d'Arduino, diguéssim que estaves fent servir la Leonardo i ara canvies a una UNO, recorda de canviar-ho també als paràmetres. Aquesta configuració es queda desada al tancar l'aplicació, així que no cal fer-ho cada cop que l'obris, només cada cop que canviïs de placa d'Arduino.

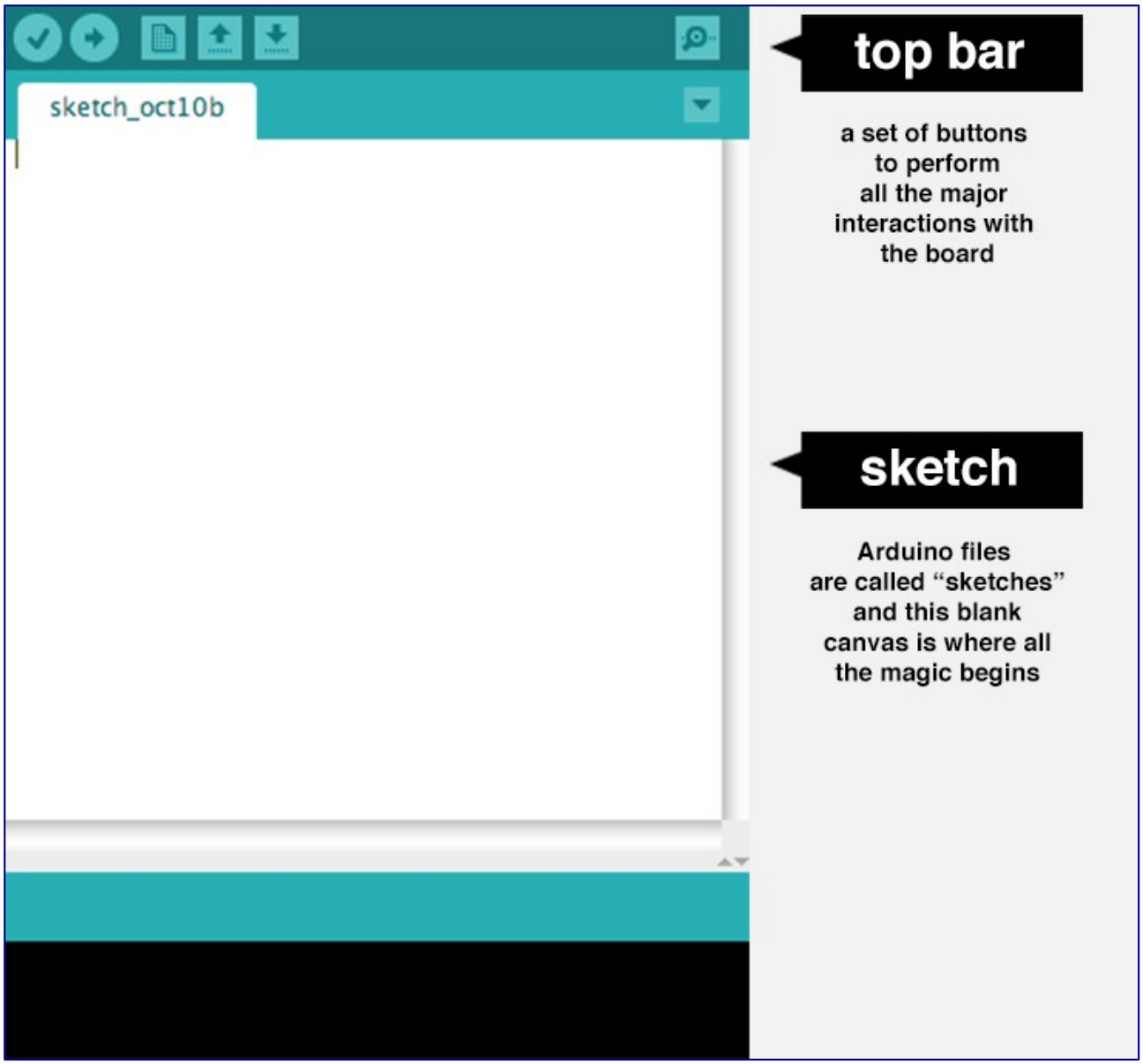


Això és tot, ja estem preparats per començar. En el següent pas començarem amb el codi!

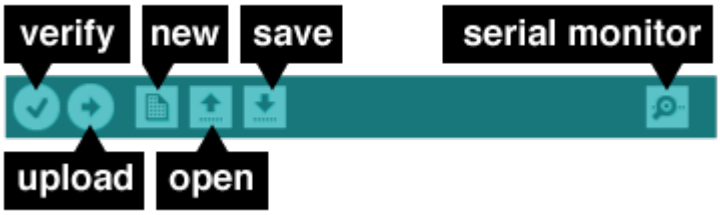
L'IDE d'Arduino

En aquest tutorial farem una introducció a l'IDE d'Arduino.

Això és el que veiem quan obrim el programa:



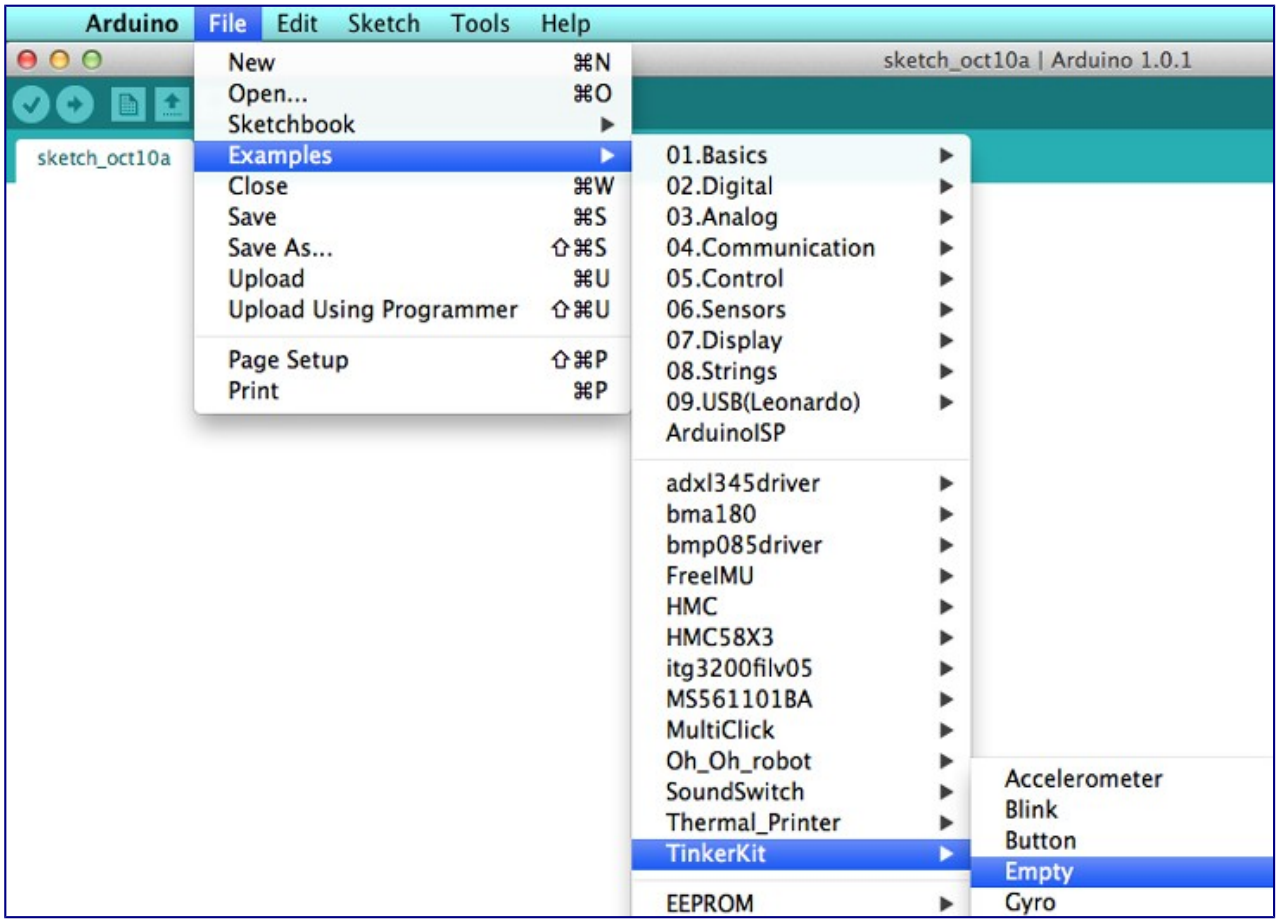
1. Barra superior

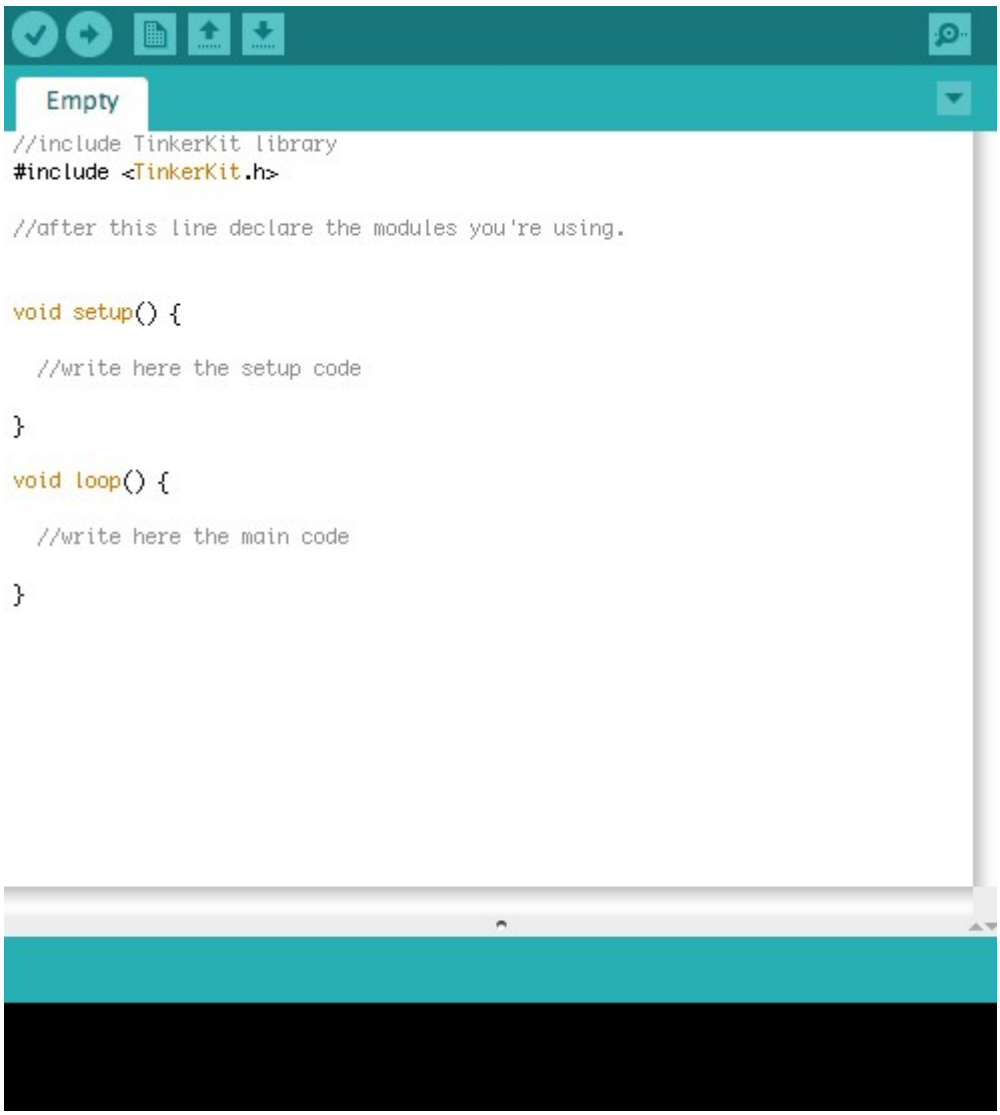


- **Comprova**: Revisar els errors en el codi
- **Carrega**: enviar el codi a la placa connectada.
- **Nou**: obre un fitxer en blanc
- **Obrir**: obrir un fitxer d'arduino (.ino)
- **Desa**: Desa el fitxer actual
- **Monitor sèrie**: obrir la finestra del Monitor en sèrie (ho expliquem més endavant).

2. {codi}

Per començar amb el codi obrim un exemple buit de **Fitxer > Exemples > TinkerKit > Empty**





Aquest és l'aspecte d'un esborrany (sketch) de codi de TinkerKit. Si ja has treballat amb Arduino, veuràs que és molt similar, exepte per la inclusió de la llibreria en la primera línia.

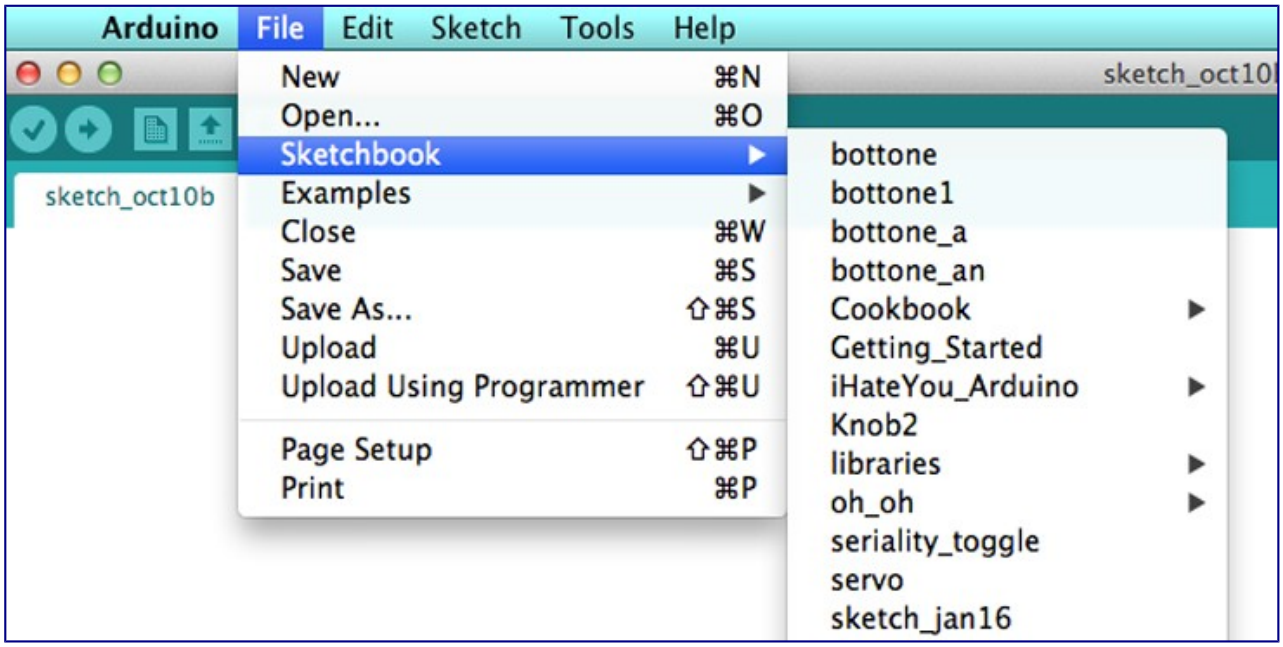
Com pots veure hi ha dues grans àrees en cada sketch de codi, anomenades **setup** i **loop.**

En l'apartat **setup** escrivim totes les coses que necessitem definir abans que comenci el nostre codi.

Tot el que s'escrigui dintre del **loop**, com el seu nom indica, es repetirà una vegada i una altra mentre l'Arduino estigui engegat.

L'exemple **Empty** és una bona manera d'iniciar els nostres projectes enlloc d'escriure el mateix codi cada vegada. Recorda que **els exemples són fitxers de només lectura**, i no poden ser modificats. Clica a Desa per desar-ho com a un projecte nou (després del avís de només lectura).

Tots esl projectes que facis amb l'IDE d'Arduino es desen per defecte en la carpeta "Arduino" dintre de la carpeta "Documents". És bastant senzill que aquesta carpeta quedi ràpidament plena de projectes. Podràs accedir-hi fàcilment als teus projectes a **Fitxer > Sketchbook**



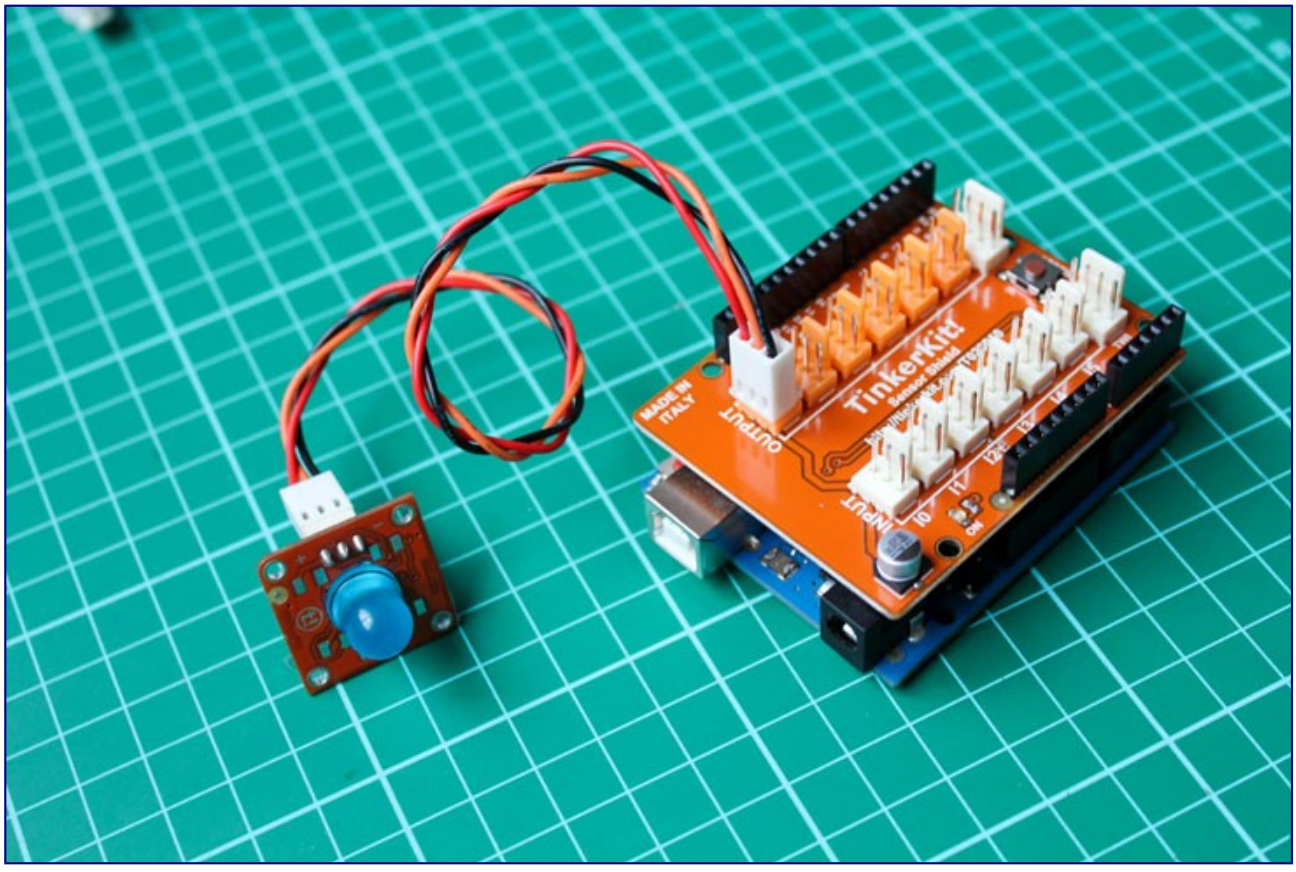
Això és gairebé tot el que necessites saber per començar el teu projecte amb l'IDE d'Arduino. En el pròxim tutorial començarem a escriure codi!

Comencem amb el codi

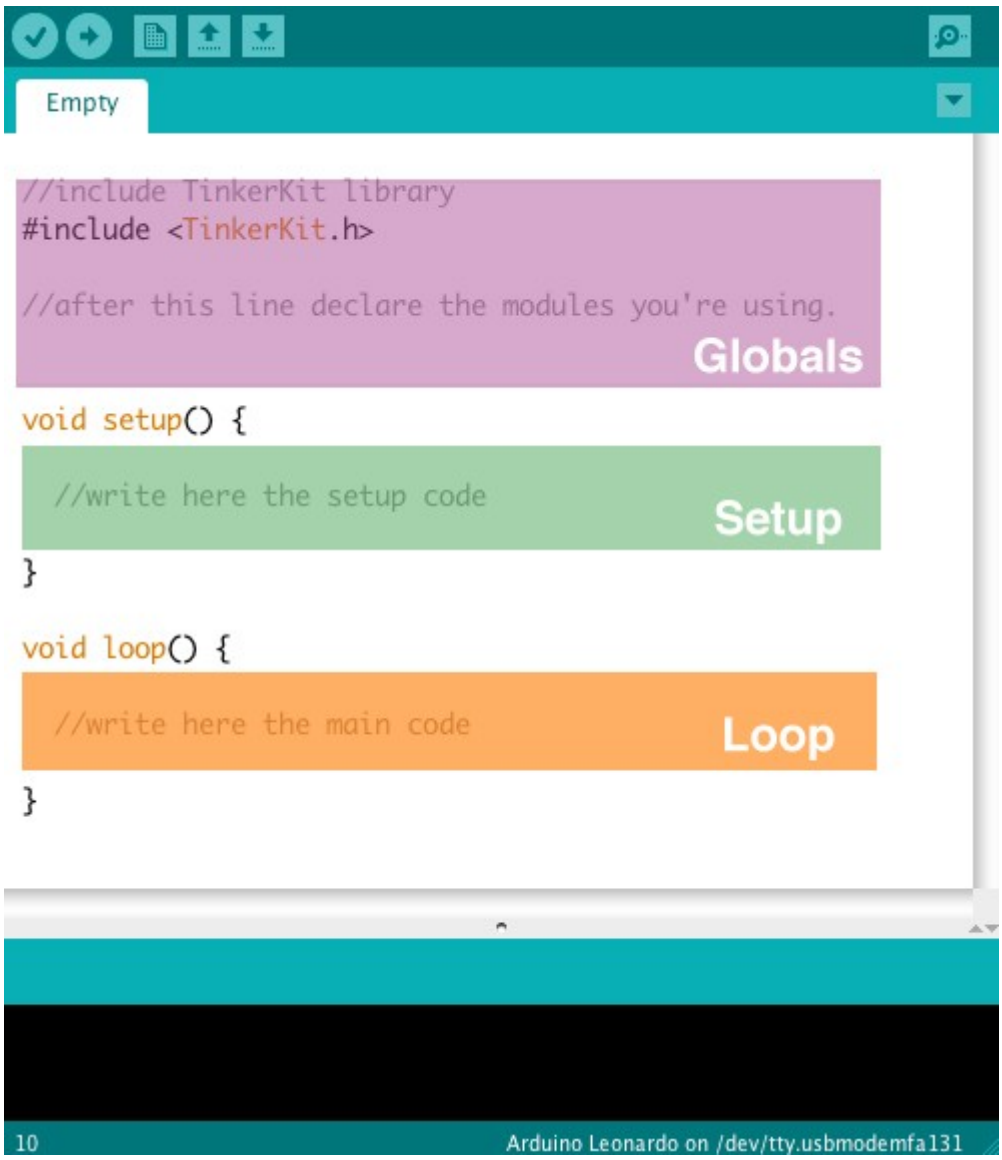
En aquest tutorial introduirem els conceptes bàsics d'uns sketch de TinkerKit. Si estàs familiaritzat amb el codi d'Arduino, aquest tutorial et serà ben senzill. Començarem una altra vegada amb un LED, però el procés és el mateix per a cada output. Els LEDs són els dispositius de sortida més senzills, en aquest tutorial ens centrarem en el codi.

Encaixa el TinkerKit com ja hem après i obre l'IDE d'Arduino, aleshores conecta un LED al port **O0**.

L'aspecte serà el següent:



Ara, en el programa, obre l'exemple Empty (en aquest punt, ja hauries de saber obrir exemples, però per si no ho recordes, el trobaràs a **Fitxer > Exemples > TinkerKit > Empty**).



Ok, aquest és l'aspecte d'un exemple buit, el punt de partida del nostre codi TinkerKit!. Està compost per tres parts: **globals**,**setup** i **loop**.

- **Globals**: És l'inici de cada sketch, aquí declarem els components i altres elements (com les variables) que farem servir en el nostre codi.
- **void setup()** (o simplement setup): aquesta funció s'executa una sola vegada al inici del nostre codi, i es fa servir per definir tot **abans de començar el loop**.
- **void loop()** (o loop): és on escrivim el codi principal. La funció loop es repeteix una vegada i una altra mentre l'Arduino estigui connectat. Totes les instruccions escrites dintre les claus **{ }** són executades de manera seqüencial, una línia darrere d'altre.

Si vols, esobrra les línies comentades, comencen amb **//**. Aquestes línies no són llegides

pel programa i s'acostumen a fer servir per explicar parts del nostre codi.

La línia que hem d'escrire en els "globals" és la següent:

```
TKLed led(O0);
```

- La **primera paraula** de la línia és la **classe**, com que hem connectat un LED hem de definir la classe **TKLed**. Si ho canviessim a un relé hauriem de definir-ho com a TKRelay, etc...
- El **segon** element és el **nom** que assignem a aquest LED en particular. En aquest exemple l'anomenem senzillament "led", si estiguéssim treballant amb diferents LEDs podríem anomenar-los led1, led2, etc. Cada component que fem servir ha de tenir un nom diferent.
- El **tercer** element és senzill, és el **número del port** on connectem el mòdul a la placa, si no n'estàs segur, mira el número de port on el cable està endollat a la shield.
- Totes les línies de codi que escrivim **han d'acabar amb un punt i coma ;** és l'error més habitual per a principiants i per a no tant principiants. Al principi és normal que se'ns oblidi, no et preocupis si no et surt de manera automàtica!

Deixem el **void setup() buit**, ja que els objectes de TinkerKit no necessiten ser inicialitzats.

Engegeum un LED escrivint dintre dels claudators { } del **void loop()**

```
led.on();
```

És senzill, no? Primer escrivim el nom del element que volem manipular, en aquest exemple, el LED que previament hem anomenat "led"; després escrivim el mètode que volem aplicar, en aquest cas, encendre'l, és a dir ".on()". Això **encén el LED** a la seva brillantor màxima. No oblidis el **punt i coma al final** de la línia i pujar-ho tot a la placa fent servir el botó de **Puja**. El teu LED s'hauria d'encendre, sino, revisa el codi complet per trobar-ne l'error:

```
#include <TinkerKit.h>
TKLed led(O0);
```

```
void setup() {
}

void loop() {
  led.on();
}
```

Ara prova d'escriure .off() on hi havia .on(), i puja de nou l'sketch a la placa per veure que passa: el teu LED hauria d'apagar-se.

Ok, ara fem que parpellegi. El que farem, en altres paraules, és encendre el LED, esperar un segon, apagar-lo i esperar un altre segon. Per fer això, haurem de fer servir una funció que s'anomena **delay**. El delay "congela" la placa per el temps que definim, expressat en mili-segons (1 segon = 1000 mili-segons). Així és com li diem a la placa que "esperi".

```
delay(1000);
```

La línia de sobre pot ser llegida com "**espera un segon**" (1000 mili-segons).

En la placa d'Arduino les instruccions s'executen una després de l'altra, així que el nostre loop hauria de ser així:

```
led.on();
delay(1000);
led.off();
delay(1000);
```

Puja-ho a la placa per veure com el LED parpalleja.

El parpelleig d'un LED és el "Hola Món" de la computació física, cada component té les seves classes i mètodes però més o menys aquesta és la dinàmica del codi de TinkerKit:

- Declara els components en els globals,
- assigna mètodes en la funció loop, i
- puja el codi

I ARA?

Comença a jugar amb altres components, en el pròxim tutorial farem servir un sensor per controlar les sortides i farem servir la condició (l'operador "if").

★ Repte ★

Prova de deixar el LED a mitja brillantor, en comptes d'encendre'l a màxima potència.

La solució, en el pròxim tutorial!

Sensors i Actuadors

Al final del anterior tutorial t'hem proposat un repte: prova d'encentre el LES a mitja brillantor. Si has buscat la referència, la solució és prou intuïtiva, la classe TKLed té un mètode anomenat .brightness() on podem escriure la brillantor que volem que tingui el nostre LED dintre del parèntesi.

Els valors que podem escriure van del **0** al **1023**, on 0 vol dir OFF i 1023 és ON, a màxima potència.

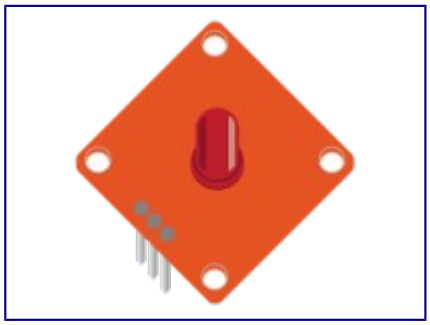
Si has fet servir Arduino abans, segurament notaràs que la sortida PWM dels TinkerKit va amb una escala de 0 a 1023 enlloc de la escala de 0 a 255 que acostumem a fer servir. Això és per que la llibreria de TinkerKit, a fi de simplificar el procés, converteix automàticament el valor que introduïm en el mètode .brightness().

Això fa que convertim el led.brightness(valor) a analogWrite(ledPin, valor/4).

De fet, el LED està sempre a la seva potència màxima, però parpelleja tant de pressa que el nostre ull humà és incapaç de percebre-ho. El resultat és que nosaltres percebim una menor brillantor. Això s'anomena PWM i és la manera de simular sortides analògiques amb senyals digitals.

Els actuadors

Per cada actuador necessitem conèixer que el valor a escriure dintre dels parèntesi del mètode, va del 0 al 1023.



Off: led.brightness(0)

Mig-On: led.brightness(512)

Complet-On: led.brightness(1023)

Els sensors

Pels sensors és una miqueta diferent. Primer, tenim dues categories principals: els sensors **digitals** i els sensors **analògics**.

Els sensors **digitals** només poden tenir dos estats, anomenats **HIGH** (alt) i **LOW** (baix). Alguns exemples són els botons, els sensors d'inclinació o els sensors de tacte. Aquests sensors només poden llegir dos valors, els botons, com els sensors de tacte, només et poden dir si estan pressionats o no. Si li escrius al botó el mètode .read(), és com si li preguntessis: "ei botó, que estàs llegint?", i la resposta seria HIGH si està pressionat o LOW si no ho està.

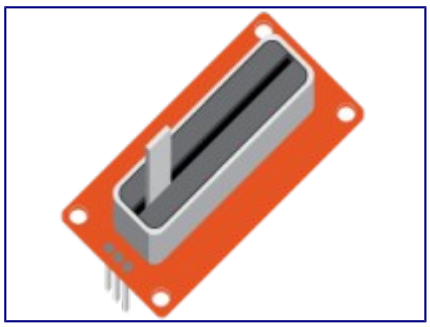
Quan NO està pressionat:

- **Codi:** button.read() **Valor retornat:** LOW
- **Codi:** button.readSwitch() **Valor retornat:** 0

Quan està PRESSIONAT:

- **Codi:** button.read() **Valor retornat:** HIGH
- **Codi:** button.readSwitch() **Valor retornat:** 1

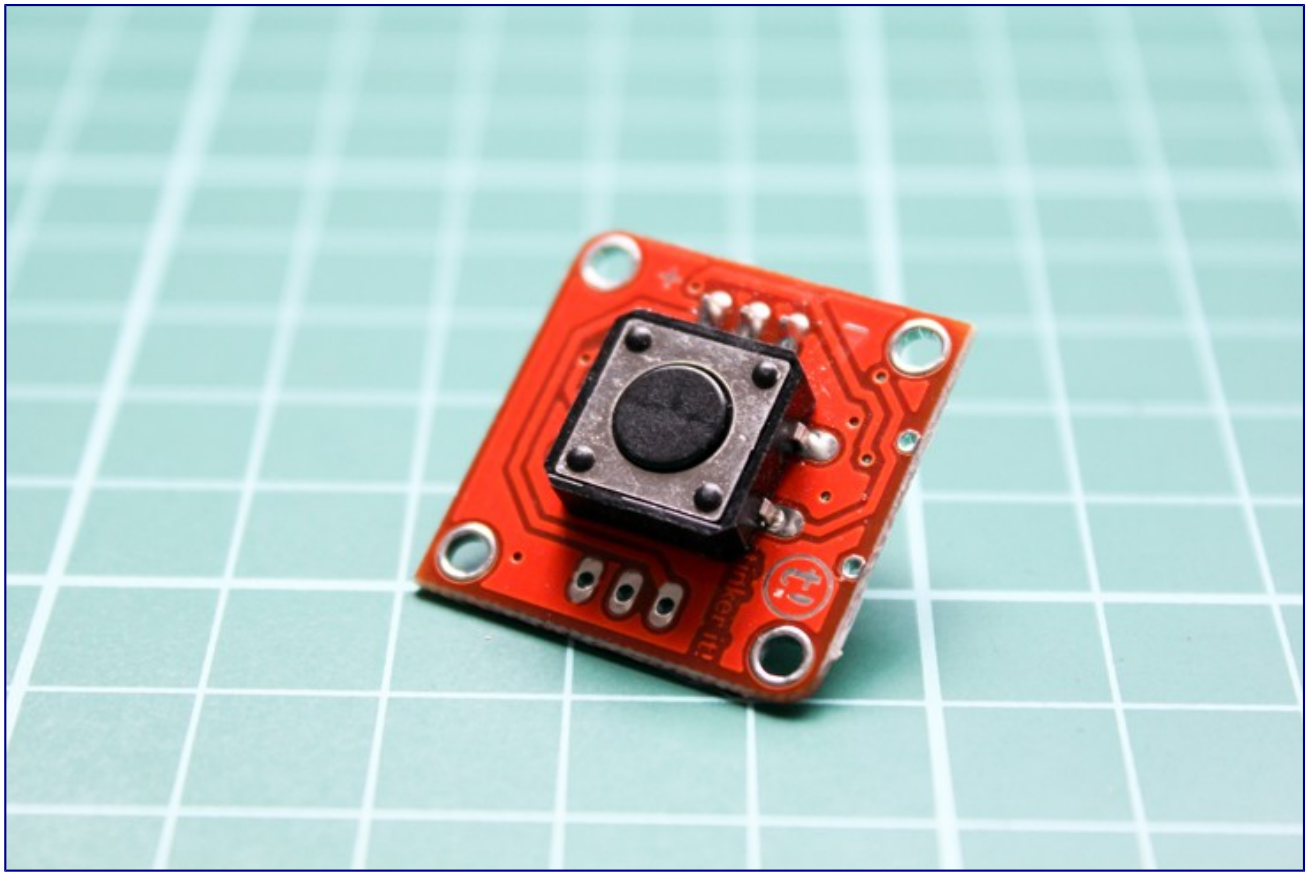
Els sensors **analògics,** per altra banda, poden llegir els valors entre mig. La majoria dels sensors TinkerKit son analògics, el seu rang de valors està **entre el 0 i el 1023**. En els exemples anteriors cridàvem el mètode .read() del botó, què passarà si cridem el mateix mètode d'un potenciòmetre o d'un sensor de llum?



- **Codi:** linpot.read() **Valor retornat:** entre 0 i 1023

El que tenim és un nombre sencer entre 0 i 1023. Si fem el mateix amb un light sensor, els valors estaran entre 0 (sense llum) a 1023 (el màxim de llum percebuda).

El botó i la condició



Comencem amb un clàssim el **botó** i el **LED**. Podem començar encenent el LES quan el botó estigui premut. Connecta el botó i el LED a la placa, un al port I0 i l'altre al O0.

El botó és un sensor digital, ens donarà només dos valors **HIGH** i **LOW**. Per saber si el botó està polsat hem de fer servir el mètode read().

Farem servir la condició lògica anomenada **IF** (si). Es fa servir per executar part del codi només si una cosa és certa. La sintaxi és prou intuïtiva, i és:

```
if (condició) {
  fes alguna cosa
}
```

Un exemple pot ser:

```
if (fa fred) {
```

```
  posa't un jersey
}
```

Però com podem dir quan fa fred? Ho hem d'escriure en nombres. Per això, tenim els operadors lògics.

>	Més gran que
<	Més petit que
==	Igual que
>=	Més gran o igual que
<=	Més petit o igual que

Per tant la funció seria:

```
if (temperatura < 17) {
  posa't un jersey
}
```

Ok, ara tornem al nostre botó. Pots desxifrar com direm "Engega el LED quan el botó estigui pressionat" en el llenguatge d'Arduino / TinkerKit? Intenta-ho escriure'l abans de continuar llegint!

La resposta és:

```
if (button.read() == HIGH) {
  led.on();
}
```

Si ho posem tot junt, amb la sintaxi correcta d'Arduino, el resultat és:

```
#include <TinkerKit.h>

TKLed led(O0);
TKButton button(I0);

void setup() {
}

void loop() {
  if (button.read() == HIGH) {
    led.on();
```

```
  }
}
```

Això és correcte, però un coo pujem el codi en el nostre TinkerKit veiem que un cop apretem el botó, el LED continua encès encara deixem de pressionar el botó. Això és per que en cap moment hem dit al LED que s'apagui. Per fer això, hem d'escriure una altra funció "if", similar a:

```
if (button.read() == HIGH) {
  led.on();
}
if (button.read() == LOW) {
  led.off();
}
```

Hi ha una manera més intel·ligent d'aconseguir aquest resultat. La funció "if" pot anar seguida d'un operador "**else**" (aleshores), que serà executat quan la condició del "if" sigui falsa. Si fem servir la funció if-else el nostre codi quedarà així:

```

if (button.read() == HIGH) {
  led.on();
} else {
  led.off();
}
```

Això funciona per què el nostre botó només té dues posicions, HIGH i LOW, pressionat o no pressionat, de manera que l'únic moment en que la condició és falsa és quan button.read() == LOW.

Fixa't que el resultat és exactament el mateix que en el exemple Button.

★ Repte ★

Abans de continuar, prova de convertir el botó en un interruptor, de manera que cada vegada que el polsem el LED canvii entre encès i apagat.

PISTA: busca a la documentació un mètode específic del botó que et puguis ervir. Si vols fer-ho pel camí difícil, sense fer servir altres mètodes del botó, necessitaràs una variable.

Llegeix els teus sensors: Serial.print();

Avui començarem a jugar amb els sensors analògics. Com hem explicat en sessions anteriors, un sensor analògic retorna un rang de valors, no com els sensors digitals, que retornes només dos valors.

Per veure el que el sensor està llegint farem servir una nova funció que s'anomena:¡

```
Serial.println();
```

Aquesta funció **escriu** sobre un port sèrie, com ara l'USB. Des de la IDE d'Arduino podem veure què està passant en el port sèrie on hem connectat l'Arduino, clicant en el botó que trobem a la part superior dreta de la nostra interfaç. Això ens obre una nova finestra anomenada **Monitor Sèrie**.

La funció serial ha de ser inicialitzada dintre del void setup(), fen servir la sintaxi:

```
Serial.begin(9600);
```

El valor 9600 és la velocitat de comunicació, expressada en Baud. És un valor fixe, generalment farem servir 9600.

Dintre de la funció loop, cada vegada que vulguem veure alguna cosa al port sèrie només hem d'escriure:

```
Serial.println("Hola!");
delay(1000);
```

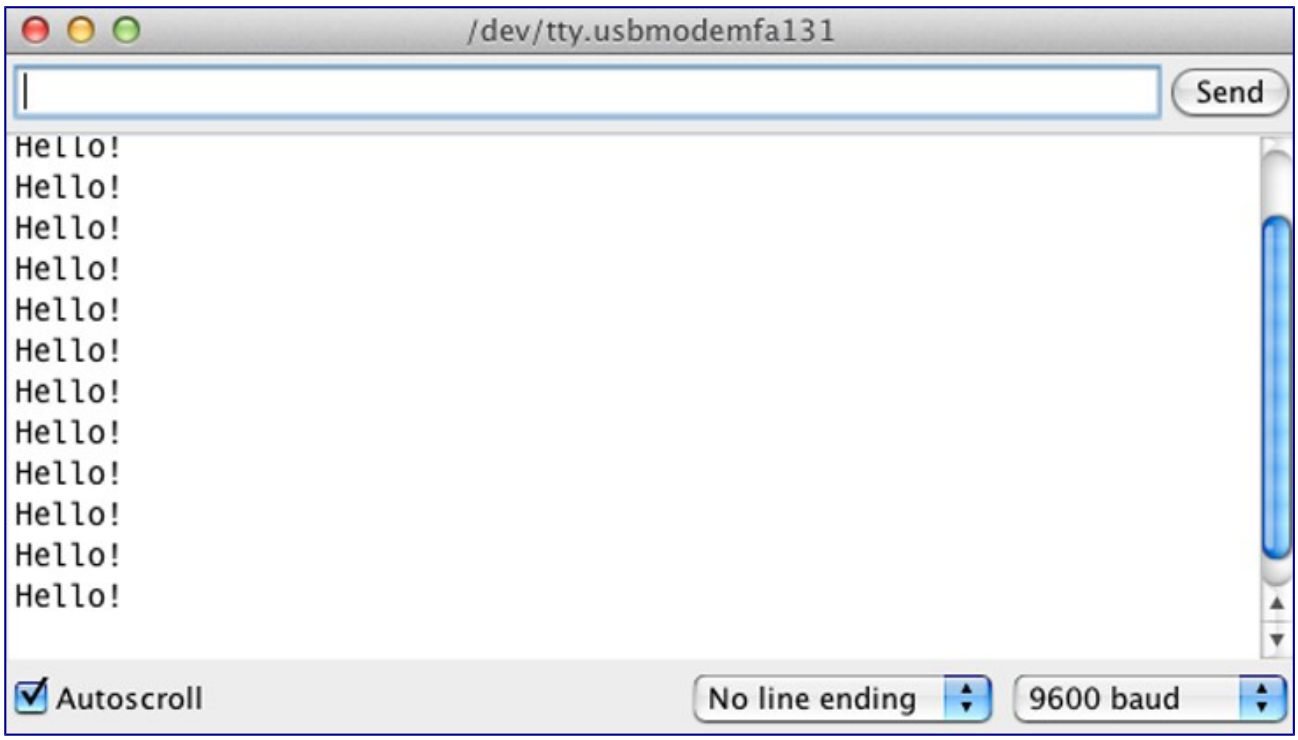
Amb això dintre del loop, el nostre programa escriurà "Hola!" en el port sèrie cada segon. Prova de pujar en el teu Arduino el codi:

```
void setup() {
```

```
  Serial.begin(9600);
}

void loop() {
  Serial.println("Hola!");
  delay(1000);
}
```

Un cop pujat el codi a la placa, obre el Monitor Sèrie, hauries de veure quelcom similar a:



Si escrius alguna cosa dintre dels parèntesi del println, ho hauràs de fer entre cometes, com hem fet amb el "Hola!", i s'escriurà exactament el que has escrit, però també podem escriure-hi un valor o una **variable**.

Què és una variable?

Les variables son la manera d'emmagatzemar valors que poden canviar amb el temps. Per crear una nova variable necessitem assignar-hi un **nom** i un **tipus**. Hi ha diferent tipus de variables, en els nostres sketches bàsics farem servir principalment valors sencers (integer), que definim com **int**.

Les variables s'han de declarar abans del setup, en la secció global. Com exemple, si volem crear una nova variable del tipus sencer, escriurem

```
int i = 0;
```

Això crea una nova variable anomenada **i** i que per ara és igual a 0. Podem incrementar el seu valor dintre de la funció loop escrivint:

```
i = i + 1;
```

Aquest codi incrementa en 1 cada vegada que s'executa la funció loop. Verifiquem-ho amb la funció Serial print que hem vist abans. En el loop escriurem Serial.print(i) i la pausa d'un segon, el nostre sketch quedarà així:

```
int i = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  i = i + 1;
  Serial.println(i);
  delay(1000);
}
```

Si obrim el monitor sèrie després de carrager aquest sketch, el que veurem és un valor que s'incrementa cada segon:

Ara, què farà la funció serial print amb els sensors analògics? Podem fer-la servir per escriure en el Monitor sèrie els valors que el sensor està llegint.

Com ho podem escriure en llenguatge d'Arduino? Intenta-ho abans de llegir la resposta!

La solució és:

```
Serial.println(sensor.read());
```

El codi sencer quedaria així:

```
#include <TinkerKit.h>
```

```
TKLightSensor sensor(I0);

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(sensor.read());
  delay(1000);
}
```

Ara que ja coneixem les **variables**, les podem fer servir per fer el nostre codi més llegible i fàcil de modificar. De fet podem enmmagatzemar el valor que ens doni senor.read() dintre de la variable:

```
#include <TinkerKit.h>
TKLightSensor sensor(I0);

int val = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = sensor.read();
  Serial.println(val);
  delay(1000);
}
```

Si fem servir el mètode **print** enlloc del **println** podem escriure en el monitor sèrie sense crear una nova línia. És útil si volem afegir detalls al valor imprès, o escriure diferents variables en la mateixa línia, només recorda de fer servir println en l'últim element.

```
Serial.print("El valor del sensor és: ");
Serial.println(val);
```

El el proper tutorial connectarem finalment els mòduls d'entrada i de sortida. Tens idea de com ho podem fer? Prova-ho tu mateix, crea un LED que reaccioni a la quantitat de llum que hi ha a l'habitació!

Connectant sensors i actuadors

En aquest tutorial finalment connectarem sensors i actuardors.

Comencem enganxant un LED al port O0 i un potenciòmetre al port I0. El codi que escrivim a continuació funciona bé amb qualsevol sensor analògic, només recorda de canviar el nom del objecte quan el declares.

El nostre codi és:

```
#include <TinkerKit.h>

TKPotentiometer pot(I0);
TKLed led (O0);

void setup() {
  // en blanc
}

void loop() {
  // en blanc
}
```

Fixem-nos en el que el nostre sensor està llegint, fent servir la funció print que hem vist en el tutorial anterior.

```
val = pot.read();
Serial.println(val);
delay(50);
```

El codi que tenim serà similar a aquest:

```
#include <TinkerKit.h>

TKPotentiometer pot(I0);
TKLed led (O0);

int val;

void setup() {
```

```
  Serial.begin(9600);
}

void loop() {
  val = pot.read();
  Serial.println(val);
  delay(50);
}
```

Un cop hem pujat aquest codi al Arduino, podem llegir els valors del potenciòmetre en el port sèrie, com hem fet abans. Fins aquí, tot be, però per què hem connectat un LED?

Farem que el LED respongui al nostre sensor, de manera que el brillo canvii proporcionalment a les variacions del nostre sensor. Pots esbrinar com ho hauràs d'escriure?

Intenta-ho tu mateix abans de continuar llegint!

Només hem de posar els valors del potenciòmetre dintre de la funció brightness del LED, en el nostre cas hem vinculat aquests valors a la variable val:

```
led.brightness(val);
```

Escriu això dintre de la funció loop i puja-ho a la placa, si mous el potenciòmetre hauries de vuere com la brillantor del LED canvia. Obre el monitor sèrie per veure el valor exacte.

Codi final:

```
#include <TinkerKit.h>

TKPotentiometer pot(I0);
TKLed led (O0);

int val;

void setup() {
  Serial.begin(9600);
}

void loop() {
  val = pot.read();
  led.brightness(val);
```

```
  Serial.println(val);
  delay(50);
}
```

Si has fet servir Arduino abans, provablement hauràs notat que la sortida PWM del TinkerKit dona un valor entre 0 i 1023 enlloc del habitual 0 a 255. Això és per que la llibreria TinkerKit, a fi de simplificar el procés, automàticament converteix el valor en el mètode .brightness(). Això converteix led.brigthness(valor) a analogWrite(ledPin, value/4).

Els estats del botó

La classe TKButton inclou una sèrie de mètodes que detecten diferents estats d'un botó. Es fan servir per veure si un botó està pressionat(**pressed**), es deixa anar (**released**), es manté premut (**held**) o s'intercanvia el seu estat (**toggled**).

Hi ha un exemple que ve amb la llibreria anomenat ButtonStates. Obrim-lo per veure com funciona.

El codi és bastant senzill, bàsicament tenim un conjunt de mètodes que són veritat (true) quan es produeix aquella acció. Per verificar-los, fem servir l'operador "if" (si). Per comprovar si el botó està pressionat, escrivim:

```
if (btn.pressed()) {
    // fes alguna cosa
}
```

Aquest codi executa les instruccions que escrivim dintre dels claudàtors només en el moment que el botó està pressionat (o és alliberat).

Un mètode interessant és el **getSwitch()**, que canvia entre 0 i 1 cada vegada que el botó és alliberat. D'aquesta manera és molt senzill fer servir un botó com un interruptor. Mirem l'expemple:

```
if (btn.getSwitch() == 0)
    led.on();
else
    led.off();
```

L'últim mètode que hem fet servir en aquest tutorial és **.held()**, només retorna veritat quan el botó està pressionat més de **500 mili-segons** (mig segon). Pots encendre el LED només quan es mantingui premut fent servir el codi:

```
if (btn.held())
    led.on();
else
    led.off();
```

El bucle *for*

En aquest apartat introduirem el bucle *for*.

Hem après a encendre un LED variant-ne la intensitat fent servir el mètode .brightness, prenent com a valors el rang entre 0 i 1023, de manera que si volem el nostre LED a mitja brillantor, escriurem led.brightness(512).

Com ho podem fer si volem que el valor de llum canvii tota la estona? Per exemple, si volem un LED amb lluminositat intermitent? Farem servir una funció que ens ajudarà, el **bucle for**.

```
for (int i = 0; i < 1023; i++) {
   led.brightness(i);
   delay(5);
}
```

La finalitat del bucle for és repetir una part de codi sempre que la condició sigui veritat. Com podem veure en l'exemple escrit, la primera part del for està escrita entre parèntesis, i es divideix en tres parts:

- **int i = 0** primer assignem el valor 0 a una nova variable anomenada **i**
- **i < 1023** mentre aquesta condició sigui veritat, el loop s'executarà
- **i++** és el mateix que i = i + 1, i incrementa el valor de i en 1

Resumint, hem creat una nova variable a la que li sumem 1 cada vegada que s'executa el loop, fins que aquesta no sigui més gran de 1023. Quan s'arriba en aquest punt, el programa segueix endavant.

La segona part del loop està dintre dels claudàtors i és la part de codi que volem executar.

```
led.brightness(i);
delay(5);
```

Dintre del nostre codi escrivim la variable **i**, que va de 0 a 1023, en el mètode brightness del LED. Hem de posar un **delay** després, si no no podríem percebre la variació de llum ja que el loop s'executa molt de pressa.

Aquest és el codi:

```
#include <TinkerKit.h>

TKLedled(O0);

voidsetup() {
  //nothing
}

voidloop() {
  for(inti = 0; i < 1023; i++) {
    led.brightness(i);
    delay(5);
  }
}
```

Ara tenim un LED bategant, com ho fem per que s'esvaeixi? Hem vist com incrementar-ne la intensitat fins a tenir-lo completament brillant, però ara volem que s'esvaeixi lentament fins que s'apagui.

Saps com fer-ho? Intenta-ho abans de continuar llegint la solució.

Fet? Només havies de copiar el bucle i canviar uns petits detalls. Abans era:

```
for (int i = 0; i < 1023; i++) {
    led.brightness(i);
    delay(5);
}
```

Ara és:

```
for (int i = 1023; i > 0; i--) {
    led.brightness(i);
    delay(5);
}
```

Només has de posar un bucle darrera del altre, d'aquesta manera el LED s'encén lentament i s'apaga repetidament.

```
#include <TinkerKit.h>

TKLed led(O0);
```

```
void setup() {
  //nothing
}

void loop() {
  for (int i = 0; i < 1023; i++) {
    led.brightness(i);
    delay(5);
  }
  for (int i = 1023; i > 0; i--) {
    led.brightness(i);
    delay(5);
  }
}
```

També podem jugar amb la velocitat de tot aquest procés d'esvaniment canviant el valor del delay. Per fer-lo ràpid, podem fer servir una variable que emmagatzemi el valor de manera que no haurem de canviar-ho en els dos bucles cada vegada:

```
#include <TinkerKit.h>

TKLed led(O0);
int del = 5;

void setup() {
  //nothing
}

void loop() {
  for (int i = 0; i < 1023; i++) {
    led.brightness(i);
    delay(del);
  }
  for (int i = 1023; i > 0; i--) {
    led.brightness(i);
    delay(del);
  }
}
```

★ Repte ★ El semàfor.

És el moment d'un petit repte, i posem en joc tot el que hem après amb aquest tutorial bàsic: construïm un semàfor!

Algunes pistes: comença amb tres LEDs que simulin el semàfor per a cotxes, afegeix després un semàfor per peatons. Aquest últim estarà sempre apagat fins que algú vulgui creuar la carretera, i demani pas amb la llum verda, fent servir l'input que escullis. Intenta esbrinar quin sensor fas servir pel teu projecte!

Quan la llum dels peatons estigui encesa, la llum dels cotxes ha de ser vermella. Passat un cert temps, la llum dels peatons s'ha d'apagar, i les llums dels altres semàfors han de tornar a l'activitat normal.

El semàfor (solució)

Fem servir un sensor de llum i quatre leds: tres pels cotxes i un per als peatons.

```
TKLed pedGreen(O0);
TKLed carYellow(O1);
TKLed carRed(O2);
TKLed carGreen(O3);
TKLightSensor action(I0);
```

Bàsicament hem dividit el nostre codi en dues parts: una per al tràfic de cotxes i una altra per als peatons, només s'executa quan ells sol·liciten la llum verda. Podem escriure tot el codi només fent servir un if i uns quants delays. Si els valors que es llegeixen estan dintre de cert llindar el codi de peatons s'executa. En el nostre cas el valor és 100 però cada sensor llegeix valors diferent en diferents contexts, hem de fer servir el Serial.print() per veure les lectures i escollir el nostre valor.

La llum de cotxes està sempre verda excepte que passi alguna cosa. El nostre void loop serà similar a:

```
void loop() {
    carGreen.on();
    val = action.read();

    if (val < 100) {
       // vermell per a cotxes i verd per a peatons
    }
}
```

Volem canviar la llum de tràfic a groc i després a vermell, i aleshores la llum de peatons es posi verda.

```
carGreen.off();
carYellow.on();
delay(3000);

carYellow.off();
carRed.on();
```

```
pedGreen.on();
delay(4000);
```

És millor per als peatons que els avisem abans que la llum s'apagui, de manera que es puguin donar-se pressa per creuar. Només tenim un LED, així que farem que parpallejar.

```
for(int i=0; i<7; i++) {
  pedGreen.off();
  delay(200);
  pedGreen.on();
  delay(200);
}
```

Després de la llum parpallejant, apaguem tots els LEDs:

```
carRed.off();
pedGreen.off();
```

Aquest és el nostre codi.

Veiem com ens ha quedat!

```
#include <TinkerKit.h>

TKLed carRed(O2);
TKLed carYellow(O1);
TKLed carGreen(O3);

TKLed pedGreen(O0);

TKLightSensor action(I0);
int val;

void setup() {
  Serial.begin(9600);
}

void loop() {
  carGreen.on();

  val = action.read();
```

```
  if (val < 100) {

    carGreen.off();
    carYellow.on();
    delay(3000);

    carYellow.off();
    carRed.on();
    pedGreen.on();
    delay(4000);

    for(int i=0; i<7; i++) {
    pedGreen.off();
    delay(200);
    pedGreen.on();
    delay(200);
    }

 carRed.off();
pedGreen.off();
 }
}
```