ROBÒTICA · 3r ESO

UNITATS 1-2: COMPUTACIÓ FÍSICA · INTRODUCCIÓ A ARDUINO · L'IDE

Nom: _____ Curs/Grup: _____ Data: _____

Setmanes 1-2. Al final sabràs:

- Explicar què és la computació física (sensors → microcontrolador → actuadors).
- Identificar la placa Arduino UNO i els seus components.
- Instal·lar i fer servir l'IDE d'Arduino (verificar, carregar, monitor sèrie).
- Pujar el teu primer programa (Blink) a la placa.

1. Computació física amb Arduino

La computació física consisteix a programar objectes físics perquè interactuïn amb el món real. El flux de treball de tot sistema de control és:

ENTRADA
Sensor (llum, pulsació, distància...)

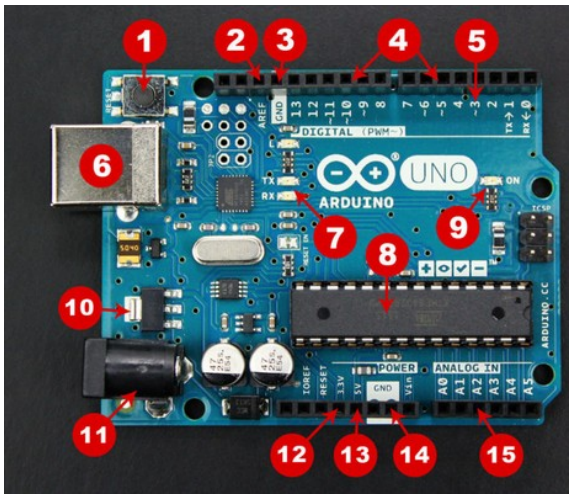
> *OBJECTIUS DEL CURS: fonaments de pensament computacional · programació textual (C++ amb l'IDE d'Arduino) · entrades i sortides digitals · entrades analògiques i sortides PWM · servomotors · comunicació sèrie.*

2. Què és Arduino?

> *DEFINICIÓ: La placa Arduino és un dispositiu electrònic open-source basat en un microcontrolador programable que permet llegir entrades (llum ambiental, pulsació d'un botó, presència d'un objecte...), processar aquesta informació i produir una sortida (moviment d'un motor, encesa d'un LED, so...).*

Arduino és present en milions de projectes: des de robots educatius i projectes escolars fins a microsatèl·lits, impressores 3D i sistemes de control industrials. El

seu èxit ve de popularitzar tecnologies que abans només entenien els enginyers, gràcies a la comunitat open-source.



1- Botó de Reset – Això reiniciarà qualsevol codi que estigui carregat a la placa Arduino 2
2- Pin AREF – Significa "Analog Reference" (Referència Analògica) i s'utilitza per establir un voltatge de referència extern
3- Pin de terra (Ground) – Hi ha uns quants pins de terra a l'Arduino i tots funcionen igual
4- Entrada/Sortida digital – Els pins 0-13 es poden utilitzar per a entrada o sortida digital
5- PWM – Els pins marcats amb el símbol (~) poden simular una sortida analògica
6- Connexió USB – S'utilitza per alimentar el teu Arduino i carregar programes (sketches)
7- TX/RX – LEDs indicadors de transmissió i recepció de dades
8- Microcontrolador ATmega – Aquest és el cervell i és on s'emmagatzemen els programes
9- LED indicador d'engegada (Power) – Aquest LED s'il·lumina sempre que la placa està endollada a una font d'alimentació
10- Regulador de voltatge – Això controla la quantitat de voltatge que entra a la placa Arduino
11- Connector d'alimentació DC (Power Barrel Jack) – S'utilitza per alimentar el teu Arduino amb una font d'alimentació externa
12- Pin de 3.3V – Aquest pin subministra 3.3 volts de potència als teus projectes
13- Pin de 5V – Aquest pin subministra 5 volts de potència als teus projectes
14- Pins de terra (Ground) – Hi ha uns quants pins de terra a l'Arduino i tots funcionen igual
15- Pins analògics – Aquests pins poden llegir el senyal d'un sensor analògic i convertir-lo a digital

La placa Arduino UNO amb els seus components.

La placa pot programar-se en diversos llenguatges

- TEXTUALS: C amb l'IDE d'Arduino (el que usarem en aquest curs).
- GRÀFICS: per blocs, com versions de Scratch.

Alimentació

- USB: proporciona 5 V (la via habitual quan programa).
- JACK d'alimentació: pila de 9 V o font recomanada entre 7 i 12 V.

Pins d'alimentació (per alimentar el circuit del breadboard)

Pin
3.3 V
5 V

GND
Vin

Pins d'entrada i sortida

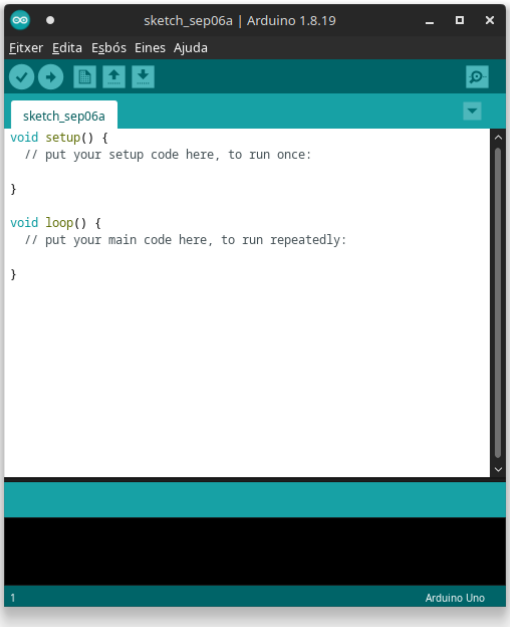
- DIGITAL: sortida 0 V (LOW) o 5 V (HIGH); entrada: 0-2 V es llegeix LOW i 3-5 V HIGH.
- SORTIDA ANALÒGICA: de 0 a 5 V en 256 valors intermedis (precisió de 8 bits). En realitat és un PWM (senyal modulada per amplada de pols).
- ENTRADA ANALÒGICA: de 0 a 5 V en 1024 valors (precisió de 10 bits).
- La intensitat màxima per pin és de 40 mA.

El circuit es munta sobre una placa de prototips (BREADBOARD o PROTOBOARD) i les connexions es fan amb cables JUMPER (no es trenquen als sòcols).

3. Programant Arduino: estructura bàsica

Tot programa Arduino té 3 parts:

Part
1. Declarar variables: Serveix per crear i donar un nom a espais de memòria on guardarem dades (com nombres o estats) que el programa farà servir i podrà modificar.
2. void setup(): És una funció que s'executa **només una vegada** quan s'encén o es reinicia l'Arduino. S'utilitza per configurar aspectes inicials, com dir si un pin funcionarà com a entrada o com a sortida.
3. void loop(): És una funció que s'executa **contínuament en bucle** (un cop rere l'altre, de manera infinita) un cop acaba el setup(). És on va el codi principal que controla el comportament de l'Arduino.



Nomenclatura de variables

Els noms de variables NO poden començar per número ni portar símbols o espais. Noms vàlids: ledPin, estatAnterior, comptaPulsacions, mevaVariable, lecturaSensor...

A la programació d'Arduino (que està basada en C++), existeixen diferents **tipus de variables** segons la dimensió i el tipus de dada que necessitis guardar. Utilitzar el tipus adequat t'ajuda a estalviar memòria a la placa.

Aquí tens els principals tipus de variables que s'utilitzen habitualment:

- **int (Enter):** Guarda nombres enters, tant positius com negatius, des de −32.768 fins a 32.767. Ocupa 2 bytes.
    - *Exemple:* int comptador = 150;
- **float (Decimal):** S'utilitza per guardar nombres amb decimals. Ocupa 4 bytes i pot contenir valors molt grans o petits.
    - *Exemple:* float temperatura = 23.5;
- **bool o boolean (Booleà):** Només pot tenir dos valors possibles: true (cert/1) o false (fals/0). Ideal per a estats com encès/apagat o obert/tancat.
    - *Exemple:* bool estatPolsador = true;
- **char (Caràcter):** Emmagatzema un sol caràcter (una lletra o símbol) utilitzant cometes senzilles. Ocupa 1 byte.
    - *Exemple:* char lletra = 'A';
- **byte:** Guarda nombres enters sense signe, des de 0 fins a 255. Ocupa 1 byte, ideal per estalviar memòria.
    - *Exemple:* byte lluminositat = 200;
- **long (Enter llarg):** Per a nombres enters molt més grans, des de −2.147.483.648 fins a 2.147.483.647. Ocupa 4 bytes (molt útil, per exemple, per mesurar temps en microsegons amb millis()).
    - *Exemple:* long temps = 123456789;
- **String (Cadena de text):** S'utilitza per guardar frases o paraules senceres (text). Va entre cometes dobles.
    - *Exemple:* String missatge = "Hola, mon!";

Activitat 1

Identifica quins noms són vàlids i quins no: 2led, mi_led, led pin, ledPin, contador#1. Corregeix els invàlids.

4. Instal·lació de l'IDE d'Arduino

Passos per instal·lar-lo al teu ordinador:

- 1. Visita https://www.arduino.cc/en/software

- 2. Selecciona l'última versió per al teu sistema operatiu i descarga el paquet.
- 3. Obre'l i segueix l'assistent d'instal·lació (accepta permisos i termes).
- 4. Confirma instal·lar TOT, inclosos els drivers de les placas Arduino.
- 5. Accepta la carpeta proposada per defecte i prem Instalar.
- 6. Quan acabi, prem Cerrar i obre l'IDE des de la icona.

5. Usant l'IDE: els 6 icones bàsics

Icona
VERIFICAR
CARREGAR
NOU
OBRIR
GUARDAR
MONITOR SÈRIE

Localitza'ls al teu programa.

Menús importants

- Archivo → Ejemplos: col·lecció de sketches inclosos que ensenyen tots els comandos (organitzats per temes). MOLT ÚTIL.
- Herramientas → Tarjeta: selecciona la placa que uses (Arduino UNO: ATmega328P, 16 MHz, 6 entrades analògiques, 14 E/S digitals, 6 PWM).
- Herramientas → Puerto: selecciona el port on està connectada la placa (/dev/ttyACMx a Linux, COM4+ a Windows).
- Sketch → Incluir biblioteca: afegeix llibreries (#include) al començament del codi.
- Herramientas → Formato automático: dona format llegible al programa (sangries).

Activitat 1

Obre l'IDE i identifica els 6 icones. Obre Archivo → Ejemplos → Basics i mira quants exemples hi ha. Apunta'n 3 que et cridin l'atenció:

1. _____ 2. _____ 3. _____

5. PROPOSTA 1 · El teu primer programa: Blink

Segueix aquest passos que ara anirem desglossant pas a pas per comprovar que tot funciona:

- 1. Obre l'IDE des de l'escriptori.
- 2. Connecta la placa per USB (si és la primera vegada, pot demanar permís per instal·lar drivers).
- 3. Herramientas → Tarjetas: selecciona Arduino UNO (o Arduino BT si uses la BQ ZUM).
- 4. Herramientas → Puerto: selecciona el port de la placa (TRUC: desconecta-la i mira quin port desapareix).
- 5. Archivo → Ejemplos → Basics → Blink.
- 6. Prem CARREGAR. Durant la pujada els LEDs Tx i Rx parpellegen. Al final: 'Subido' + memòria usada.

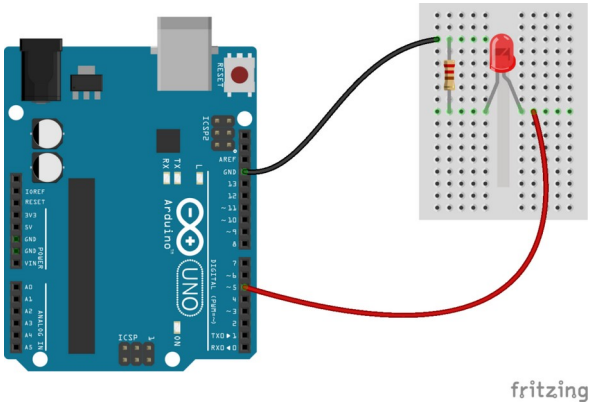
La sortida amb digitalWrite

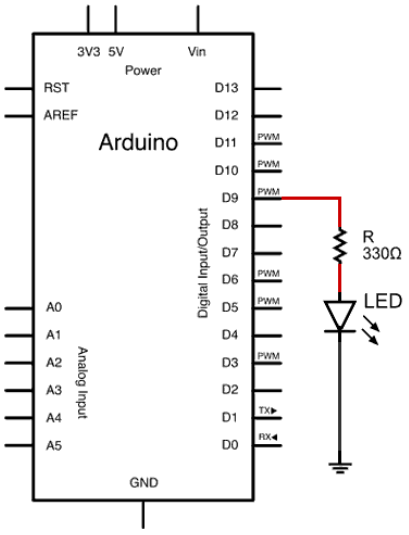
Una sortida digital només val 0 V (LOW) o 5 V (HIGH). La funció és:

> *digitalWrite(pin, HIGH); // encén · digitalWrite(pin, LOW); // apaga*

El cablejat a la placa de prototips i l'esquema

⚠ Sempre que connectem un LED cal posar una RESISTÈNCIA en sèrie (entre 100 Ω i 1 kΩ) per evitar que es cremi.





Muntatge del parpadeig: LED, resistència i GND al breadboard. Cablejat: LED al pin 13 (fila vermella), resistència 220 Ω, tornada a GND.

Pràctica 1: Parpelleig d'un LED

- MUNTATGE: LED al pin 13 + resistència 220 Ω a GND.
- PROGRAMA: setup → pinMode(13, OUTPUT); loop → digitalWrite HIGH, delay(1000), LOW, delay(1000).

S’expressa així al programa IDE:

El codi de Blink

```
void setup() {
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```

Aquest programa encén el LED integrat al pin 13 i el fa parpellejar 1 segon indefinidament. Observa l'estructura de qualsevol sketch:

- void setup(): s'executa UNA vegada en engegar (configuració).
- void loop(): es repeteix indefinidament (el programa).
- El programa es guarda en memòria FLASH: no s'esborra en desconnectar. Cada vegada que alimentis la placa, s'iniciarà automàticament.
- ⚠ La placa només pot emmagatzemar UN programa: si en puges un de nou, esborres l'anterior.

  **Observa que no es declaren variables perquè aquí no n’hi ha. Pregunta’t què és una variable... altre cop per reafirmar el concepte.*

Activitat 2 — Modifica el Blink

Puja el Blink original i comprova. Després modifica el codi:

- a) Canvia els delays a 200 ms → què passa?
- b) Fes que el LED estigui 3 segons encès i 0,5 apagat.
- c) EXTRA: fes un patró tipus 'tick-tick' (2 curtes + 1 llarga).

Prova
a
b
c

Pràctica 2: Llum que avança a través de 5 LEDs

5 LEDs alineats que s'encenen en seqüència. Fa servir un bucle *for* per recórrer els pins.

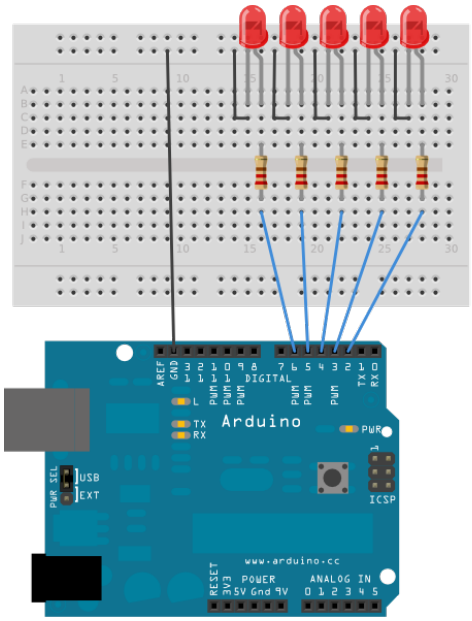
Codi esquelet que posaríem al *void loop()*:

```
for (int i = 4; i < 9; i++) {
digitalWrite(i, HIGH);
delay(100);
digitalWrite(i, LOW);
}
```

Escriu el codi que hem de posar a *void setup()* per configurar els pins de sortida dels 5 leds.

Registre

Pràctica
Parpadeig
5 LEDs en seqüència



Com que només s'encén un LED cada vegada, es podria fer el muntatge amb una sola resistència?

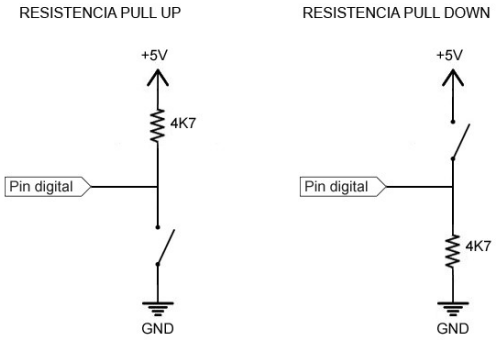
Muntatge dels 5 leds.

6. Entrades digitals: digitalRead i el pulsador

Per llegir un botó cal configurar el pin com a INPUT i fer servir una RESISTÈNCIA

DE DRENATGE (pull-up o pull-down) perquè la lectura no floti.

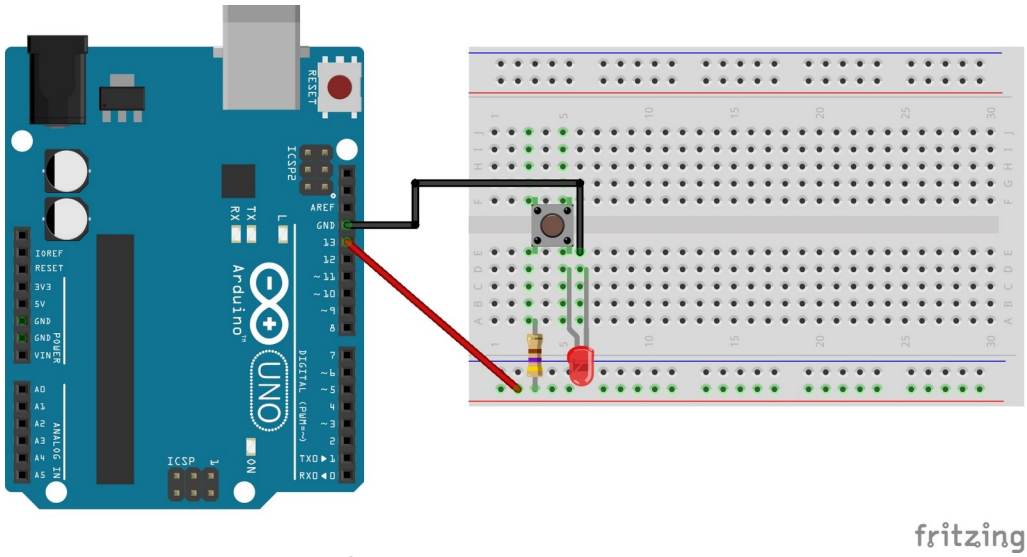
> *digitalRead(pin) retorna HIGH (5 V) o LOW (0 V) segons l'estat del botó.*



Pull-up – En prémer el posador el pin de la placa rep 0v.

Pull-down - En prémer el posador el pin de la placa rep 5v.

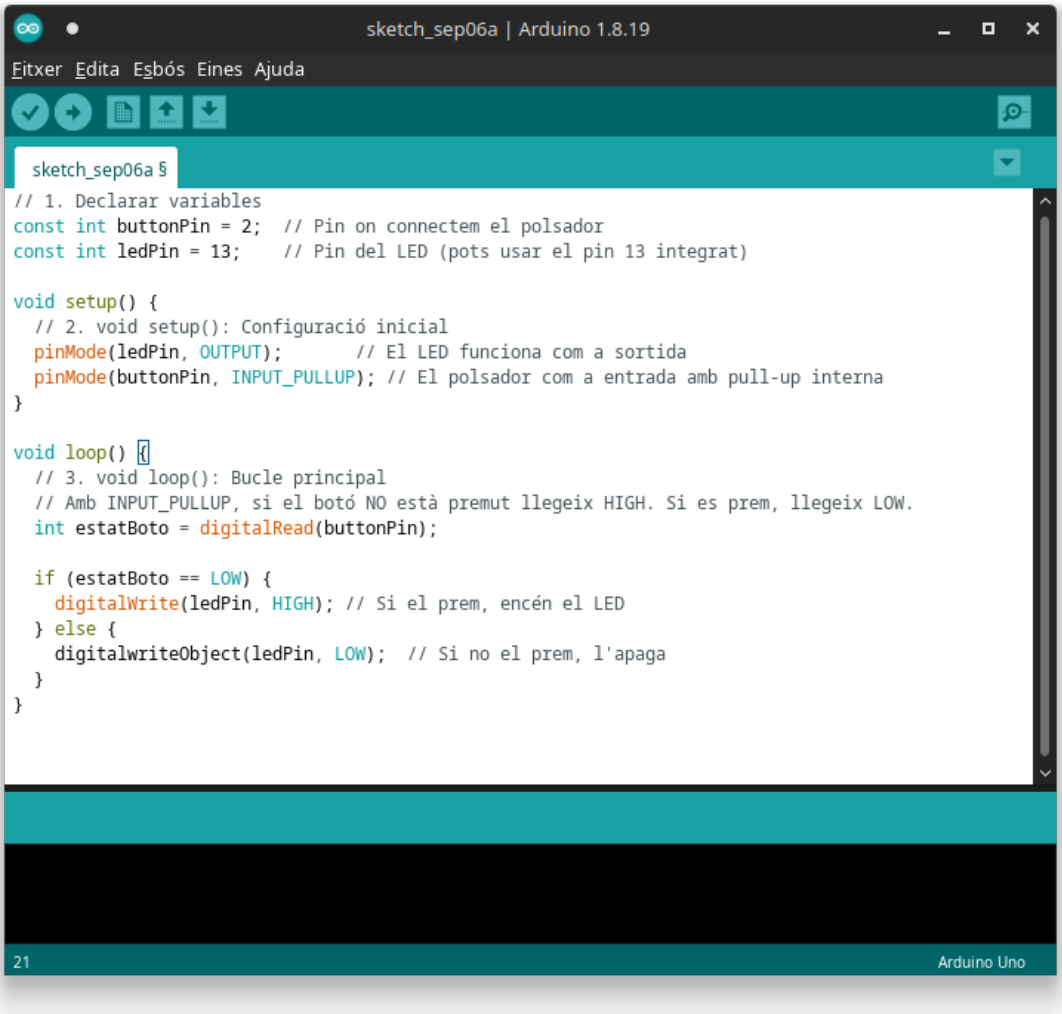
Muntatge del polsador amb la resistència de drenatge.



Muntatge del polsador amb la resistència de drenatge a la placa.

Pràctica 3: Canviar l'estat d'un LED amb un polsador

- VARIANT A: mentre estigui premut, el LED encès (lectura directa).



- VARIANT B: cada pulsació CANVIA l'estat del LED (compta pulsacions i alterna).

Per aconseguir que **cada vegada que premis el polsador** el LED canviï d'estat (s'encengui si estava apagat, o s'apagui si estava encès), necessitem detectar el moment exacte en què el botó passa de no estar premut a estar premut (l'anomenat *front de pujada* o canvi de flanc).

Si no féssim això, el programa canviaria l'estat desenes de vegades per segon mentre mantinguéssis el dit a sobre del botó.

Aquí tens el codi:

Quina diferència hi ha al codi entre la variant A i la B?

Registre

Pràctica

Pulsador variant A
Pulsador variant B

7. Atzar: random

La funció random() s'utilitza per generar nombres aleatoris (a l'atzar), la qual cosa és molt útil per a projectes com jocs, efectes de llums imprevisibles o simulacions.

Sintaxi principal:

- random(max): Genera un nombre enter entre 0 i max - 1.
- random(min, max): Genera un nombre enter entre el min i max - 1.

Exemple pràctic:

> *// Dins del loop() o d'una funció:*
>
> *int tempsEspera = random(100, 1000); // Genera un nombre aleatori entre 100 i 999 mil·lisegons*
>
> *delay(tempsEspera);randomSeed(analogRead(A0)); // inicialitza el generador aleatori*
>
> *random(min, max); // retorna un número aleatori entre min i max-1*

Pràctica 4: LED imitant una espelmaa

Un LED groc/taronja amb brillantor i temps ALEATORIS imita el parpelleig d'una flama. Fa servir PWM (analogWrite) + random.

Aquest és un codi que pots fer servir:

El LED espelma amb valors aleatoris.

Repàs i autoavaluació

Rúbrica: 1 = No assolit · 2 = Satisfactori · 3 = Notable · 4 = Excel·lent

Objectiu

Identifico els elements de la placa UNO i les seves límits (volts, mA)
Expliqu l'estructura variables/setup/loop d'un programa Arduino
Controlo sortides digitals amb digitalWrite i LEDs amb resistència
Llegeixo entrades digitals amb pulsador i resistència de drenaje
He fet les 4 pràctiques del bloc i funcionen
He construït el semàfor de la producció

Glossari

- Computació física: programar objectes físics per interactuar amb el món.
- Sensor: element que llegeix el món (entrada).
- Actuador: element que actua (sortida: LED, motor, so).
- IDE: entorn de desenvolupament integrat (programari per programar).
- Sketch: programa d'Arduino (.ino).
- Verificar / Carregar: comprovar el codi / compilar i pujar-lo.
- Monitor sèrie: finestra de comunicació PC ↔ placa.
- Flash: memòria no volàtil: el programa no s'esborra en apagar.
- Breadboard: placa de prototips per muntar sense soldar.
- Jumper: cable de connexió per al breadboard.
- HIGH / LOW: 5 V / 0 V en una sortida digital.
- PWM: senyal modulada per amplada de pols (simula analògica).
- digitalWrite / digitalRead: escriure / llegir un pin digital.
- Pull-up / pull-down: resistència de drenatge per a entrades digitals.
- random: funció d'atzar.