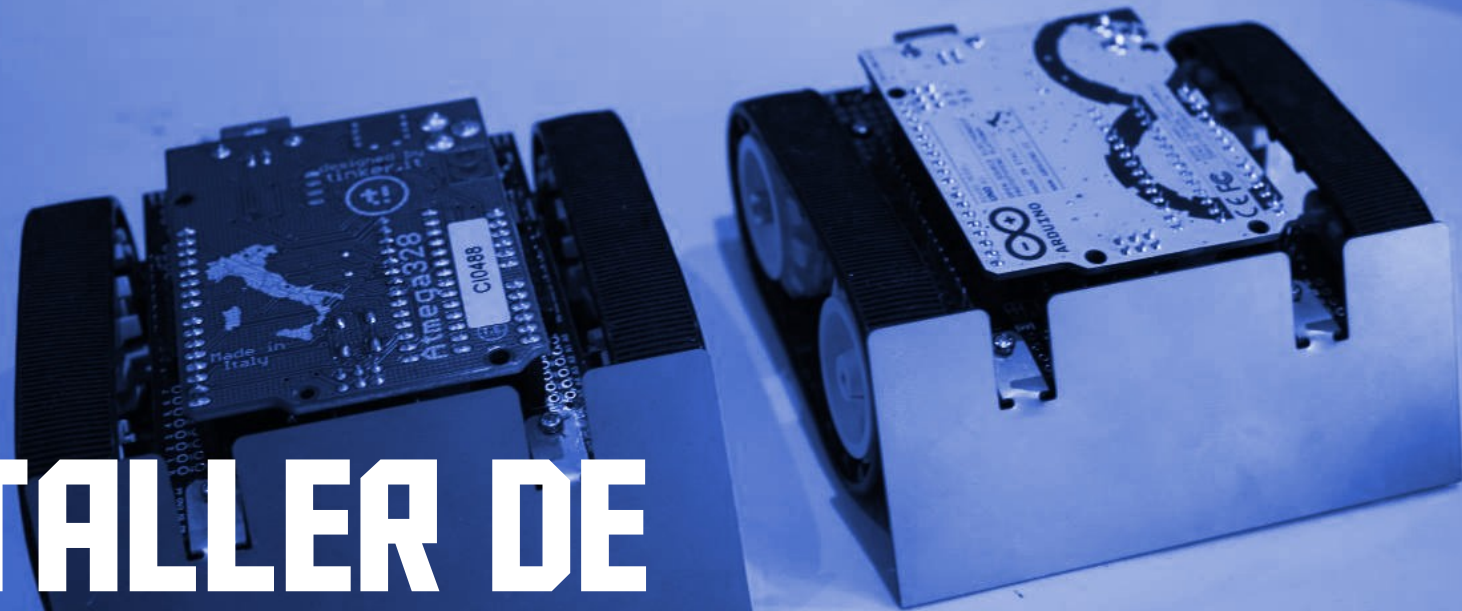


TALLER DE ROBOTICA



MUSEU DARDER
DE BANYOLES



PER ISAAC DE PALAU

Sequeix-nos a la nostra pàgina de Facebook!



Robòtica Astrobanyoles

Més informació:



www.astrobanyoles.org

Pàgina web d'Arduino



www.arduino.cc

El repte:

Any 2047. L'agència espacial de la Federació de la Terra està començant la colonització de Mart. Les naus de transport sobre la superfície del planeta no reposen ni un instant portant material de les mines d'extracció fins a la base. Però heus ací que un vehicle de transport automatitzat s'ha avariat en un lloc remot. Les sondes que orbiten el planeta l'han detectat després de fotografiar la superfície de la zona. Malauradament l'avaria és greu i no pot tornar sol.

És vital recuperar la càrrega de minerals sigui com sigui!

Els enginyers de l'agència necessiten construir un robot i programar-lo per tal que es dirigeixi fins a les coordenades que la sonda espacial els ha enviat. El robot, sense control directe humà, ha d'arribar al lloc de la nau avariada sense incidents.

Per acabar-ho de complicar, els models meteorològics preveuen una gran tempesta de pols a la zona per d'aquí 5 dies. Ens hem d'afanyar!

Se'n sortiran els nostres enginyers?

Què he de portar?

Cada alumne haurà de portar:



Ordinador portàtil: amb port USB i amb la IDE d'Arduino instal·lada (de seguida us explicarem com s'instal·la). El sistema operatiu és indiferent.

També és recomanable portar carregador.



Dossier: necessitem tenir-lo a mà durant tot el taller. Podem portar-lo imprès, però com que la tinta val més que la sang d'unicorn, recomano consultar-lo des de l'ordinador.



Estris per escriure: Amb un bolígraf i una llibreta n'hi haurà prou. No haurem de prendre apunts, però una llibreta ens pot anar bé per esbossar el codi d'un programa.

4 piles AA carregades per fer funcionar el robot. Si són recarregables, millor!

Recordem-nos de recollir-les al marxar... Els organitzadors no es faran responsables de cap pèrdua!



I sobretot, les ganes de passar-ho bé! La robòtica és un camp divertidíssim i entre tots hem de mantenir una actitud positiva.



La resta de material (robots, components, fils, eines...) el proporciona el Museu. Sobretot no el malmeteu i el tracteu-lo amb cura (les peces també tenen sentiments... sabíeu?) i en acabar torneu-lo tot al seu lloc.

Com instal·lar la IDE d'Arduino

Per tal de poder treballar i evitar perdre temps durant el curset tothom ha de portar el seu ordinador amb la IDE d'Arduino instal·lada. Es tracta d'un programa que ens ajudarà a programar el nostre robot. Anem a veure com es fa:



Màquina virtual JAVA:

Primer de tot és necessari descarregar i instal·lar la màquina virtual Java, si no la teniu ja en el vostre ordinador, per tal que la IDE d'Arduino funcioni. Escolliu i descarregueu la vostra versió depenent del sistema operatiu del següent enllaç: <https://www.java.com>

Nota!

Si surgeix algun problema a la hora d'instal·lar la IDE, contactar amb **Astrobanyoles**.

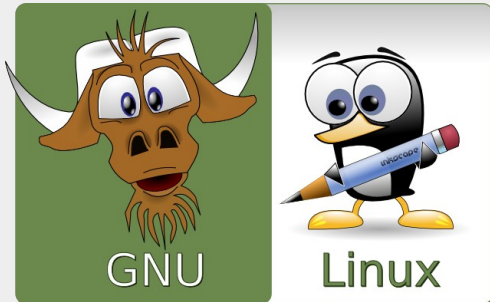
Windows:



- Descarregar l'arxiu de: <http://www.arduino.cc/en/Main/Software>
- Anar a la carpeta on s'ha descarregat i extreure'l
- Instal·lar l'executable
- La IDE hauria d'estar disponible en el menú INICIO
- En cas que no detecti la placa Arduino quan la connectem al port USB, hem d'instal·lar manualment els drivers de windows del port Serial Com. Anem a Panel de Control → Sistema i seguretat → Sistema → Administrador de dispositius → Dispositivo desconocido (si l'hagués detectat, posarà Arduino) → Premem el botó dret del ratolí per a cercar el controlador mitjançant el menú d'arxius. Cerquem dins la carpeta que hem descarregat i descomprimit d'Arduino, la carpeta que porta per nom Drivers.

Nota:

Si en lloc de la versió zip hem descarregat i instal·lat la versió instal·lable, la carpeta d'Arduino estarà segurament a c:\Archivos de programa\Arduino o similar. Tot i que el concepte és el mateix, aquest procediment per instal·lar els drivers pot variar depenent de la versió de Windows que s'utilitzi.



Linux:

Opció 1:

-Des del centre de control (Synaptic, Yast o similar, depenent de la vostra distribució) cercar el paquet «Arduino» i instal·lar-lo.

Opció 2: (només per a Debian, Ubuntu o similars)

- Obrim la Terminal i escrivim: `sudo apt-get install arduino`
- Escrivim el nostre password i s'instal·larà el programa



Mac:

- Descarregar l'arxiu de: <http://www.arduino.cc/en/Main/Software>
- Anar a la carpeta on s'ha descarregat i extreure'l
- Instal·lar l'executable

Com funciona un robot?



Els robots funcionen d'una forma semblant a nosaltres. Bé, de moment encara no he conegut cap humà que funcioni amb piles i s'hagi de programar... però sí que compartim algunes coses.

El nostres sentits són capaços de rebre estímuls del món exterior, que s'envien al cervell perquè els processi. En funció d'aquests estímuls, fem una cosa o una altra.

Per exemple, si s'està calant foc, el nostre nas detecta olor de cremat i envia una senyal al cervell. Aquest la interpreta i envia una ordre molt clara a les nostres cames...

ÉS HORA DE SORTIR CORRENT!

Els robots funcionen de forma similar: el sensor de fum detecta que s'està cremant quelcom, i envia un pols elèctric al controlador (cervell) del robot. Aquest analitza la senyal que li arriba i n'envia una altra cap als motors per sortir corrent.

Pels humans ja són innates moltes de les respostes a certes situacions. Per desgràcia, els robots no saben com han de reaccionar si no els ho diem. Per això cal programar el robot.

Programar és l'acció d'escriure un programa: una llista d'instruccions que donem al nostre robot sobre com ha d'actuar en certes condicions. És com una recepta de cuina que t'explica com fer un plat pas a pas. Per exemple, si volem que el nostre robot d'última generació ens porti el menjar a taula, haurem de programar:

```
-Agafa la safata  
-Surt de la cuina  
-Ves al menjador  
-Serveix el menjar
```

Vaja! El robot acaba de xocar amb la porta i foradar-la! Ens hem oblidat de dir-li que la obrís... pot semblar quelcom evident per a nosaltres, però als robots se'ls ha de dir tot...

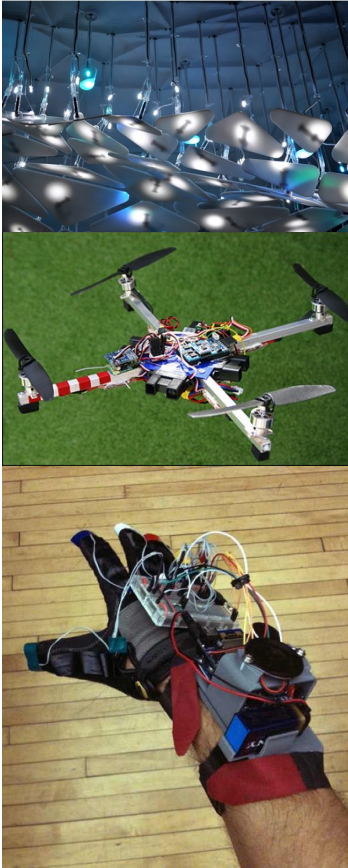
```
-Posa el menjar a la safata  
-Obre la porta  
-Surt de la cuina  
-Ves al menjador  
-Serveix el menjar
```

Nota!

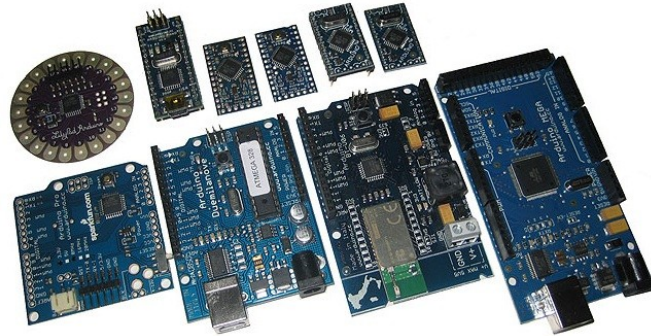
Els robots reben les instruccions en el seu propi llenguatge. En aquest cas hem posat instruccions molt genèriques que un robot no seria capaç d'entendre. Un programa seria més aviat com:

```
-Activa el motor 2  
-Espera 3 segons  
-Para el motor 2  
-Espera 3 segons  
-Llegeix el sensor 5
```

Què és Arduino?



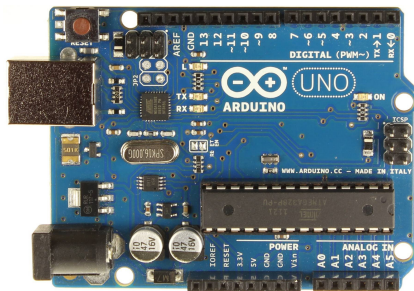
Arduino serà el cervell del nostre robot. Es tracta d'una placa pensada per fer tot tipus de projectes d'electrònica: des de drones i robots a sintetitzadors musicals i sistemes domòtics. Però això no vol dir que sigui difícil de fer funcionar! Precisament un dels punts forts de les plaques Arduino és la seva senzillesa. En alguns països nòrdics fins i tot s'ensenya a programar amb Arduino a les escoles!



Arduino és una placa de Hardware lliure. Tothom es pot descarregar els seus esquemes per construir la seva pròpia placa Arduino casolana, i fins i tot, millorar-la! I el millor és que tot aquest coneixement es pot compartir. Això ha permès que Arduino tingui una gran comunitat d'adeptes i seguidors: des d'aficionats a l'electrònica a investigadors en el camp de la robòtica, la intel·ligència artificial o la realitat augmentada.

Hi ha moltes versions de plaques Arduino diferents. Algunes són versions més avançades de plaques velles. Altres estan pensades per tasques més concretes (per exemple, Arduino Lillypad es fa servir per projectes de roba intel·ligent).

En aquest taller, farem servir Arduino UNO, la versió més estàndard.



Arduino també disposa d'una IDE especial per programar. Una IDE és com un bloc de notes que ens permet escriure programes des del nostre ordinador i carregar-los a la placa a través d'USB.

Per què Arduino?

Aquest cursat de robòtica utilitza exclusivament software i hardware lliure: Arduino i el seu IDE (entorn de programació). Per què?

Des d'Astrobanyoles promovem i advoquem per la difusió de la cultura científica i tecnològica amb els principis de llibertat englobats en l'anomenada "cultura lliure". És a dir, el coneixement compartit en benefici de la comunitat inspirat en els quatre principis del "programari lliure".

Aquests principis o llibertats essencials són els següents:

- 0** Llibertat d'utilitzar el programa per a qualsevol finalitat.
- 1** La llibertat d'estudiar com funciona el programa i adaptar-lo a les teves necessitats. L'accés al codi font és una condició prèvia d'això.
- 2** La llibertat de distribuir còpies amb les que pots ajudar als altres.
- 3** Llibertat per millorar el programa i alliberar les millores en públic, de manera que tota la comunitat se'n beneficiï. L'accés al codi font és un requisit previ per això.

Aplicant directament el mateix concepte al hardware (parts físiques d'un sistema informàtic), el hardware lliure el pots utilitzar per copiar-lo i redistribuir-lo amb o sense canvis.

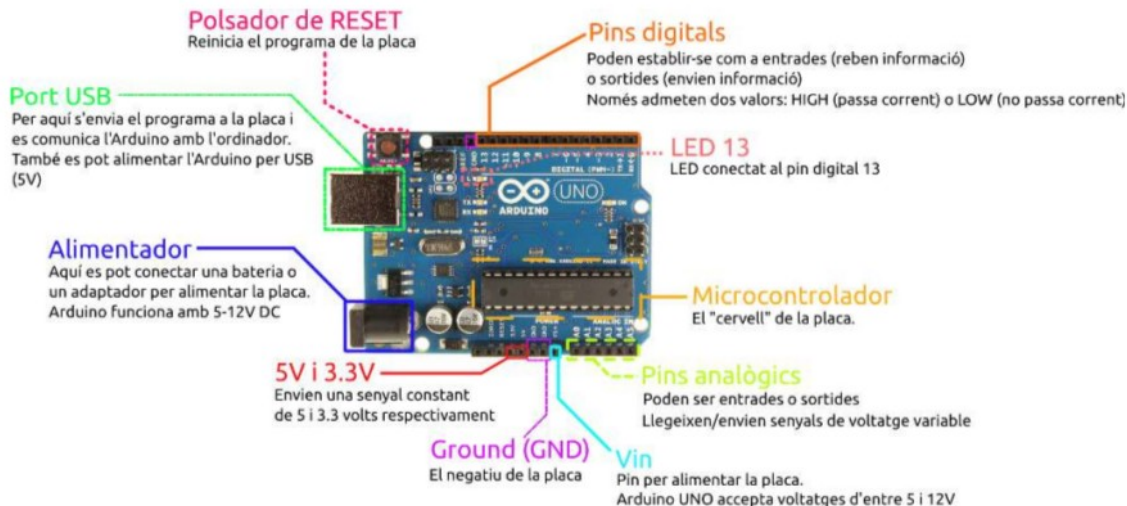
En realitat el concepte de "hardware lliure" es refereix al "disseny del hardware" (esquema dels circuits). Per tant, estem parlant d'un disseny que permeti als usuaris utilitzar-lo, copiar-lo i redistribuir-lo amb o sense canvis. Aquest disseny ha de tenir aquestes mateixes quatre llibertats definitòries del programari lliure.

En resum, "hardware lliure" té el significat de hardware amb un disseny lliure disponible.

Anatomia d'una placa Arduino

Arduino

consta de moltes parts diferents. Aquí en tenim un resum d'algunes...



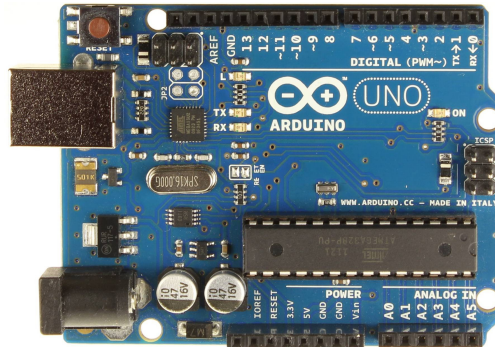
No cal que intentem recordar-les totes! Ja les anirem repassant i aprenent com funcionen al llarg d'aquest dossier.

Entrades i sortides

Tots els pins d'Arduino poden ser de dos tipus: entrades (reben informació de l'exterior) i sortides (envien informació a l'exterior)

Entrades

Reben informació de l'exterior
(sensors, botons...)



Sortides

Envien informació a l'exterior
(controlar llums, moure motors...)

Qualsevol senyal que Arduino rebí o envii pot ser de dos tipus: analògica o digital.

Una senyal digital només pot tenir dos valors: encès o apagat. Per exemple, un botó connectat a una entrada digital només tindrà dos valors (el botó està premut o no).

Una senyal analògica pot prendre infinits valors (en teoria). Per exemple, podem fer servir una senyal analògica per controlar una llum i fer que s'encengui gradualment.

Anatomia d'un programa d'Arduino

Compilar

Busca errors en el codi

Carregar

Carrega el programa a la placa Arduino per USB

Comentaris

Anotacions en el codi. El programa les ignora.

Setup

Aquí configurem els pins. Quins volem utilitzar? Volem que siguin entrades o sortides?

Bucli principal

El gruix del programa. Les instruccions que estiguin aquí dins s'executaràn indefinidament. Quan el programa arribi al final del bucli tornarà a començar.

IMPORTANT!!!!

Totes les instruccions (o la gran majoria) acaben en punt i coma al final!!!

```
Blink | Arduino 1:1.0.5+dfsg2-2
Filter Edit Sketch Help

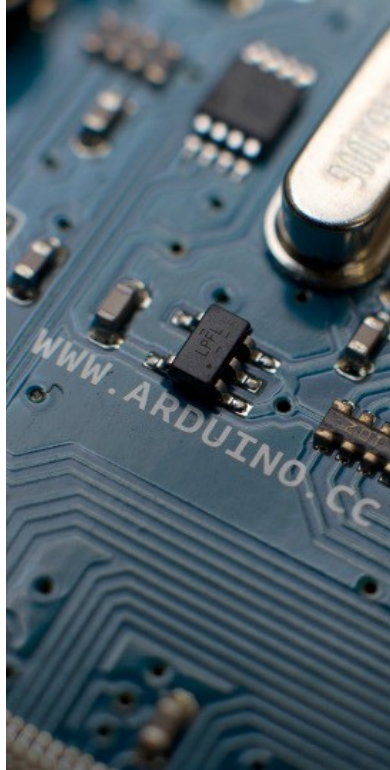
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever.
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Practica 1: BLINK

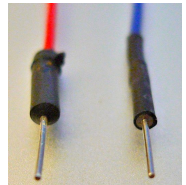


Quan un neòfit entra en el món de la programació, el primer programa que se li fa escriure és un HolaMón. Un Holamón és un programa que escriu per pantalla el text «Hola, Món!».

L'equivalent a un Holamón amb Arduino és un BLINK, un programa que encén i apaga un LED en intervals regulars de temps. Normalment es fa servir el LED que ja porta incorporada la placa en el pin D13, però nosaltres muntarem un circuit.

Què necessitem?

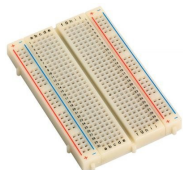
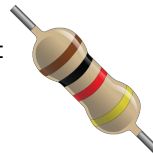
Fil elèctric: Ens ajudarà a fer les connexions entre Arduino i la Protoboard.



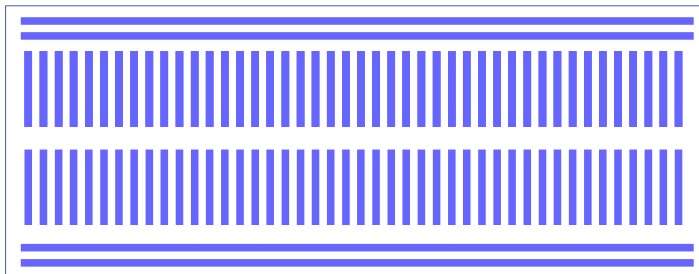


LED: els díodes són components que deixen passar el corrent elèctric només en un sentit. Un LED és un tipus de díode que, a més, fa llum.

Resistència: el corrent elèctric que proporciona Arduino és molt baix, però pot fer malbé el LED. Per tal de protegir-lo farem servir una resistència, que limitarà el pas del corrent.



Protoboard: ens ajudarà a construir el nostre circuit sense necessitat d'haver de soldar res. Es tracta d'una placa amb petits forats que estan connectats entre sí seguint aquest esquema:



Nota!

Els LEDs tenen polaritat. Això vol dir que hem de connectar sempre el positiu a la pota curta i el negatiu a la llarga, o no funcionarà!

Nota!

En cas de dubte a l'hora de muntar el circuit, demanem sempre ajuda als monitors!

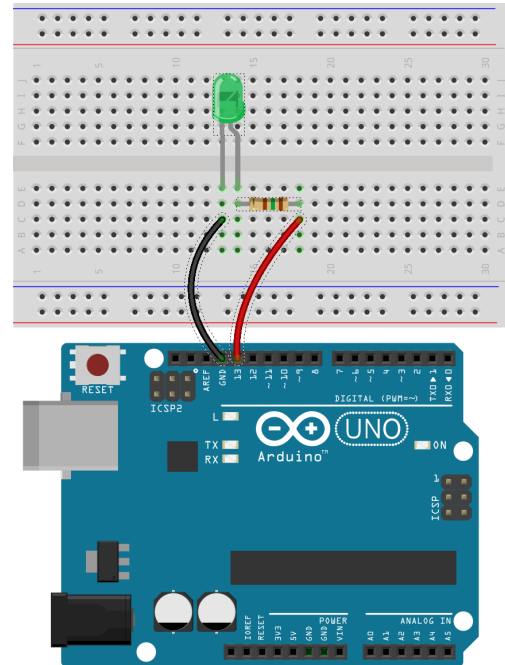
Bones praxis...

Cal ser ordenat a l'hora de muntar circuits!
Mantinguem sempre un codi de color pels fils:

- Vermell pel positiu
- Negre pel negatiu
- Altres colors per les connexions amb Arduino

El circuit:

L'ànode del LED (pota llarga) anirà connectada al pin D13 (pin digital núm. 13) d'Arduino (protegida per una resistència). El càtode, que és la pota curta, anirà connectada a GND.



Codi del programa:

Què volem fer?

-Hem connectat el LED al pin D13. Aleshores haurem d'establir aquest pin com a OUTPUT (sortida), perquè volem que es pugui enviar corrent a través d'aquest pin cap al LED.

-Volem encendre i apagar el LED en intervals d'un segon. En el bucle principal (void loop) activarem el pin 13, esperarem un segon, l'apagarem i esperarem un altre segon.

```
void setup()
{
    pinMode(13, OUTPUT); //Establim el pin 13 com a sortida
}
void loop()
{
    digitalWrite(13, HIGH); //Posem el pin 13 a HIGH

    delay(1000); //Esperem 1000 milisegons (1000 ms = 1 segon)

    digitalWrite(13, LOW); //Posem el pin 13 a LOW

    delay(1000);
}
```

Bones praxis...

Abans de començar a escriure un programa, és bo escriure en un paper el que volem fer.

Per carregar el programa, connectem la placa Arduino a l'ordinador per USB. Després, premem el botó de compilar  i el de carregar . Si tot ha anat bé, veurem que el LED s'encén i s'apaga en intervals d'un segon.

Com ho fariem per...

1. ...canviar el temps a mig segon? (500 ms)
2. ...encendre dos LEDs a la vegada?
3. ...encendre tres LEDs un rere l'altre?

ERRORS!

La màquina és perfecte, però per desgràcia l'home no. És possible que al intentar carregar el codi del programa surti un missatge d'error, o que el LED no s'encengui com volem. Com a futurs constructors de robots, ens hem d'acostumar a tractar amb errors. Solen ser de tres tipus:

Errors de compilació: Un dels errors més freqüents és escriure malament alguna instrucció. Al compilar, surt un missatge d'error si quelcom està mal escrit.

Com solucionar-los? La IDE d'Arduino ens dirà on està l'error de sintaxi i què falta. És fàcil d'arreglar.

Errors d'execució: La creu i passió de tot programador són els errors en temps d'execució. Molt sovint els programes estan ben escrits, però ens hem equivocat a l'hora d'estructurar el codi. Per exemple: si en el programa del BLINK! ens haguéssim equivocat escrivint `delay(1)` i no `delay(1000)`, el codi compilaria bé, però no faria el que volem.

Com solucionar-los? Aquest tipus d'errors són els més difícils de solucionar. S'ha de revisar l'estructura del programa, intentant endevinar què falla.

Nota!

Aprendre a arreglar errors és tot un art. Al principi ens costarà localitzar-los, però amb pràctica serem capaços de trobar-los abans i tot de carregar el codi!

Nota!

Cal anar en compte amb els errors de hardware. Un fil mal connectat pot fer malbé algun component!

Errors de Hardware: Moltes vegades ens equivocarem a l'hora de fer les connexions i és per això que cal revisar-les bé abans d'enregar el programa. Una connexió en un pin incorrecte pot fer malbé de forma irreversible la placa o algun dels components. I no volem això...

Com solucionar-los? Revisant les connexions. A ser possible abans de carregar el programa.

Pràctica 2: variables, constants i altres bèsties quadrúpedes



Què passaria si, a partir del codi del BLINK, volguéssim canviar el LED del pin 13 al pin 8? Podem anar línia per línia canviant tots els 13 per 8, la qual cosa no és recomanable si tenim dues-centes línies de codi...

O, per no haver de canviar-ho el pròxim cop, podem fer servir variables. Una variable és com una capseta amb un nom on hi guardem un valor. Quin valor? Això dependrà del tipus de variable. Per exemple, podem crear una variable entera de nom LED on hi guardarem el número del pin.

int led = 13;

Tipus de variable:
Ens diu quina classe de valor guardarem dins la variable (un nombre enter, un decimal, una lletra...)

Nom de la variable:
Serveix per identificar-la i cridar-la al llarg del programa.

Valor de la variable:
Les variables guarden valors. En aquest cas, la variable `led` valdrà 13.

Quan escrivim la variable per primer cop en el programa, li hem de dir el seu tipus (en aquest cas, `int`), el seu nom i, si volem, li podem assignar un valor. D'això se'n diu declarar una variable. Més endavant, quan fem servir la variable ja no haurem de posar el tipus a davant.

Hi ha molts tipus de variables. Les variables **int** guarden nombres enters. Les de tipus **char**, lletres i caràcters. Les **float** i **double** nombres decimals, etc.

No barregem xurros amb marines! Les variables d'un tipus sempre s'han de sumar amb altres del mateix tipus (o restar, multiplicar, dividir...)

Bones praxis...

Per norma general, les variables s'escriuen en minúscules: `pin`, `ledPin`, `contador`, `raigLaser`...

El circuit:

El circuit que hem de muntar és el mateix que el de la Pràctica 1: BLINK!

Codi:

Com quedaria el codi de la Pràctica 1: BLINK! fent servir variables enteres (int)?

```
int pin = 13; //Declarem la variable pin i li assignem el numero 13

void setup()
{
    pinMode(pin, OUTPUT); //Establim el pin 13 com a sortida
}
void loop()
{
    digitalWrite(pin, HIGH); //Posem el pin 13 a HIGH

    delay(1000); //Esperem 1000 milisegons (1000 ms = 1 segon)

    digitalWrite(pin, LOW); //Posem el pin 13 a LOW

    delay(1000);
}
```

Les variables poden canviar el seu valor: podem sumar-les, restar-les i fer tot tipus d'operacions i transformacions amb elles.

Podem sumar variables:

```
int a = 5;
int b = 2;
int c;

c = a+b; //Ara c valdria 7
```

Podem sumar 1 al seu valor de dues maneres diferents:

```
int a = 2;

a = a+1; //suma 1 al valor de la 'a'

a++; //suma 1 al valor de la 'a'
```

Podem dividir, multiplicar, restar...

```
int a = 5;
int b = 8;
a = a*2; //valdria 10
a++; //valdria 11
a = a+5; //valdria 16
a = a / b; //ara 'a' valdria 2
```

Nota!

Si una variable no ha de canviar el seu valor en tot el programa, és millor fer servir una constant.

Les constants es declaren:

```
#define CONST 13
```

I no poden canviar mai el seu valor. Tampoc tenen tipus (int, float...)

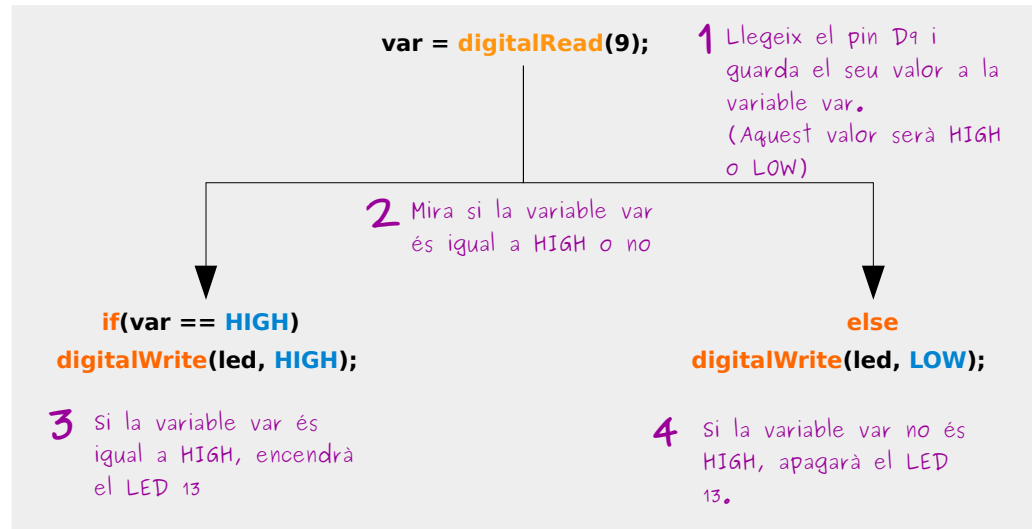
Com ho fariem per...

1. ...encendre dos LEDs posant els pins com a variables?
2. ...encendre dos LEDs posant els pins com a Constants?
3. ...fer un semàfor amb 3 LEDs? Tenint en compte que:
 - Primer s'ha d'encendre el LED verd durant cinc segons.
 - Després el LED groc durant un segon.
 - Després el LED vermell durant tres segons i mig.
 - Per últim el LED groc durant un segon.
 - I tornar a començar...
4. ...encendre i apagar un LED, de manera que el temps sigui cada vegada més llarg? (comencem amb un delay de 0ms i a cada volta del cicle es sumen 100ms);
5. ...modificar el punt 4 perquè, quan el temps arribi a 3s, torni a ser 0ms?

Pràctica 3: PEW! PEW!

En moltes ocasions, un programa ha d'escollir entre diverses formes de procedir. Per exemple: «si el robot detecta un obstacle, gira a la dreta, si no continua recte». «Si el sensor de temperatura supera els 30º, engega l'aire condicionat», etc.

Per prendre aquest tipus de decisions es fan servir els condicionals. Vegem com funcionen.



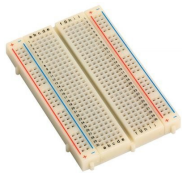
Què necessitem?

Per aquesta pràctica, necessitem un component nou:

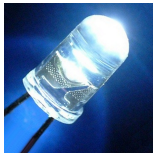


Polsador: els polsadors són petits botons que deixen passar corrent a través seu quan es premen.

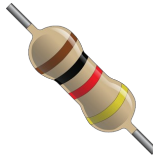
I alguns components d'altres pràctiques.



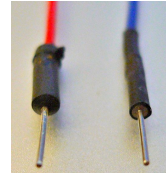
Protoboard



LED



Resistència



Fil
elèctric

Nota!

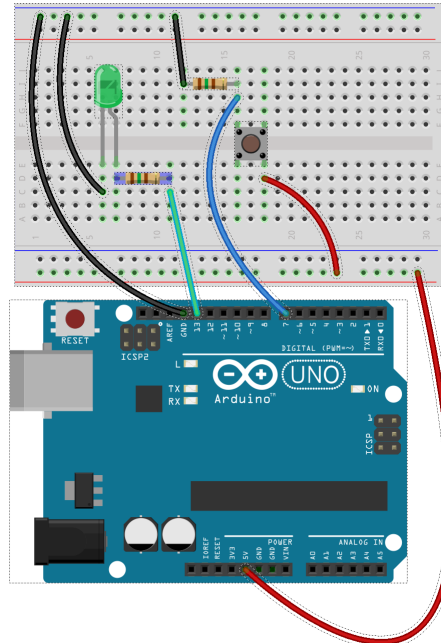
Hi ha molts tipus de polsador. Els que fem servir avui són els més estàndard, però d'altres deixen passar corrent fins que es premen, tenen més d'una posició, etc.

El circuit:

Fixem-nos en que el pulsador està connectat al pin 5V, que sempre dóna corrent. Quan premem el pulsador, aquest deixarà passar la corrent fins al pin D7 i el posarà a HIGH. Fàcil!

Nota!

Si no connectem el pulsador a GND, es produiran interferències.



Codi del programa:

La condició if mirarà si el pin D7 rep corrent (és a dir, si el pulsador està premut i per tant, el pin està a HIGH). Per veure si dues expressions són iguals, fem servir l'operador «==» (doble igual).

```
#define PULSADOR 7
#define LED 13

void setup()
{
  pinMode(LED, OUTPUT);
  pinMode(PULSADOR, INPUT); //El pin del pulsador es una entrada (INPUT)
}

void loop()
{
  if(digitalRead(PULSADOR) == HIGH) //Si el pin D7 rep corrent, encenem el LED
  {
    digitalWrite(LED, HIGH);
  }
  else //I si no, l'apaguem
  {
    digitalWrite(LED, LOW);
  }
}
```

Sobre els operadors

Hi ha molts tipus d'operadors que podem fer servir:

- < més petit
- > més gran
- <= més petit o igual
- >= més gran o igual
- != diferent

Per exemple:

```
int a = 0;
int b = 5;

if(b > a){ //Si b es mes gran que a...
    digitalWrite(13, HIGH);
}
else{
    digitalWrite(13, LOW);
}
```

També podem posar dues condicions al mateix temps. Per exemple, si volem que s'encengui un LED si es compleix que la variable `a == 2` o `a == 5`, escriurem:

```
if(a == 2 or a == 5){
    digitalWrite(13, HIGH);
}
```

De la mateixa manera, si volguéssim que s'encengués un LED si la variable `a == 14` i la variable `b == 8`, escriuríem:

```
if(a == 14 and b == 8)
{
    digitalWrite(13, HIGH);
}
```

Com ho faríem per...

1. ...afegir un segon LED controlat per un altre polsador?
2. ...crear un programa que, si es prem un botó més de 10 cops, s'encengui un LED?
3. ...fer que el LED s'encengui si es premen dos botons a la vegada?

Pràctica 4: Comunicació inalàmbrica

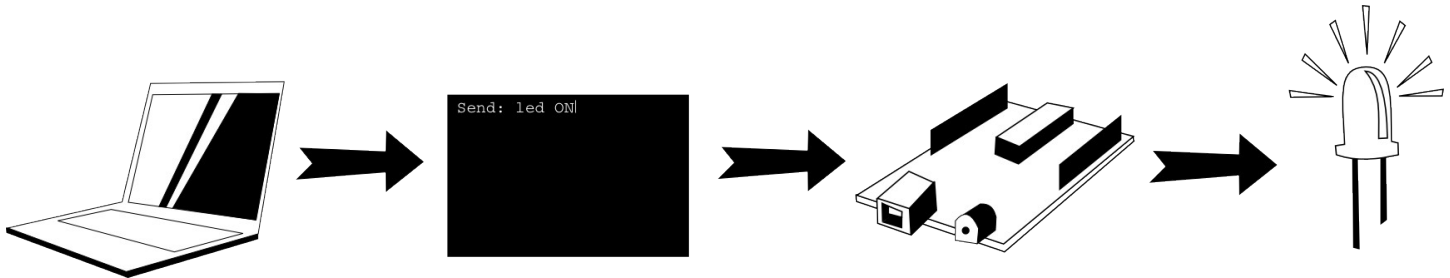
Més endavant, ens serà molt útil veure a la pantalla de l'ordinador què està pensant el nostre robot. Per exemple, ens pot interessar saber les lectures del sensor de reflectància per assegurar-nos que funcioni tot bé. De la mateixa manera, pot ser molt útil enviar instruccions al robot i, fins i tot, controlar-lo!

Nota!

Hi ha infinitat de protocols diferents: SPI, I2C, Ethernet, MIDI, PCI...

Per això es fa servir un cable USB i el protocol Serial. Un protocol és una sistema que tenen màquines diferents de comunicar-se entre sí.

En aquesta pràctica aprendrem com encendre un LED enviant instruccions des de l'ordinador a la placa, i al mateix temps, rebre missatges d'Arduino.



El circuit:

En aquesta pràctica haurem de muntar el mateix circuit que la Pràctica 1: BLINK!

Codi del programa:

En quant al programa, volem que Arduino llegeixi una lletra que li enviarem per Serial i que si es tracta d'una 'h', encengui el LED. Si es tracta d'una 'l', apagui el LED. I si no és ni una cosa ni l'altra, no faci res.

```
#define LED 13

char val; //Aquí guardarem el caracter que s'envia per serial

void setup()
{
  Serial.begin(9600); //Iniciem el Serial a 9600 baud

  pinMode(LED, OUTPUT);
}
```

```
void loop()
{

    if(Serial.available() >= 1) //Comprovem si s'esta enviant quelcom
    {
        val = Serial.read(); //Llegim el que s'esta enviant

        //Si es una 'h', encenem el LED. Si es una 'l', l'apaguem
        if(val == 'h')
        {
            digitalWrite(LED, HIGH);
        }
        else if(val == 'l')
        {
            digitalWrite(LED, LOW);
        }
        else
        {
            Serial.println("NOPE! Caracter invalid");
        }
    }
}
```

Anem a veure les instruccions noves que hem fet servir...

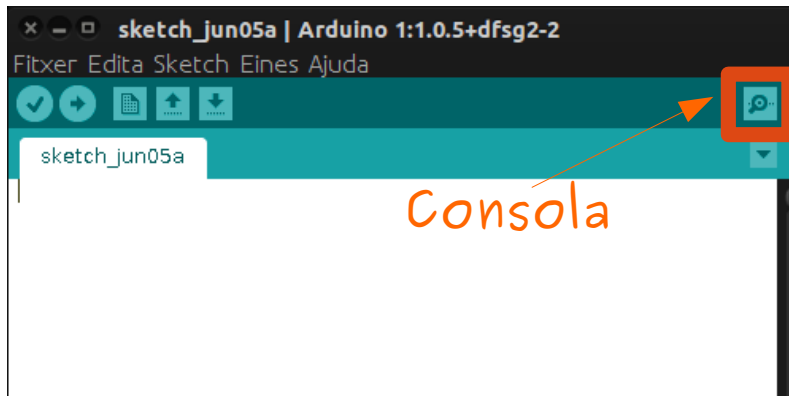
Serial.begin(baud): Obre la comunicació Serial amb l'ordinador. Baud estableix la velocitat en que es transmeten les dades (per defecte ho deixarem a 9600).

Serial.available(): mira si s'està enviant quelcom pel port de sèrie. Si no s'envia res, retorna 0.

Serial.read(): retorna el caràcter que s'ha enviat per Serial. El podem guardar dins una variable, com hem fet en el codi d'exemple.

Serial.println(): envia un missatge a través del port de sèrie, amb un salt de línia al final (un «punt i a part»).

Per veure els missatges que s'envien per Serial, hem d'obrir la consola d'Arduino:



Nota!

Si no volem un punt i a part, podem fer servir el comande `Serial.print()`. També podem escriure el salt de línia a mà amb el caracter «\n», per exemple: `Serial.print(«Hola!\n»)`

Com ho fariem per...

1. ...un programa que encengui tres LEDs diferents enviant els caràcters '1', '2', '3', i que s'apaguin tots amb el caràcter 'l'?
2. ...modificar el punt 1 perquè s'apaguin quan es torni a escriure el seu número? (exemple: si enviem '1', s'encèn el LED '1'. Si tornem a enviar '1', s'apaga).

Pràctica 5: Senyals analògiques



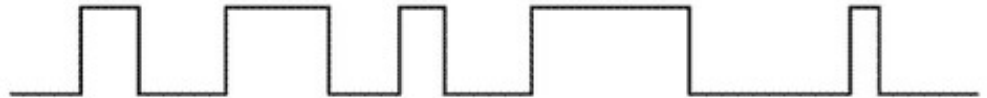
Fins ara hem treballat amb senyals digitals que només tenen dos possibles estats: HIGH o LOW. Però també existeixen les senyals analògiques.

A diferència de les senyals digitals, en que s'envia corrent o no s'envia corrent, les senyals analògiques poden prendre qualsevol valor entre la corrent mínima i la corrent màxima. Si ens fixem en la imatge de sota, veurem que la senyal analògica té molta més varietat que la digital:

Senyal analògica



Senyal Digital



Arduino pot emular senyals analògiques. Els pins A0-A5 poden llegir senyals analògiques (INPUTS), mentre que els pins 3, 5, 6, 9, 10 i 11 poden crear-ne (OUTPUTS).

En aquesta pràctica aprendrem a llegir i escriure senyals analògiques. Per això, farem servir un sensor de distància i encendrem un LED amb més o menys intensitat en funció de la distància que detecti el sensor.

Hi ha dos tipus principals de sensors de distància: per ultrasons i per infrarojos.

Els sensors d'ultrasons funcionen d'una manera semblant a com s'orienten els ratpenats: envien un soroll inaudible i mesuren quant tarda a rebotar contra un objecte i retornar.

Els sensors d'infrarojos fan servir un principi semblant, però mesurant quina quantitat de llum infraroja rebota de l'objecte. Quanta més llum, més a prop està l'objecte.

Nota!

Els pins analògics poden variar en funció del model de la placa. Aquests són per la versió UNO, la que fem servir al curset.

Què necessitem?

Per aquesta pràctica, necessitem un component nou:

Nota!

Els sensors d'infrarojos són més barats i fàcils d'usar que els d'ultrasons. Per contra, no són tant precisos.



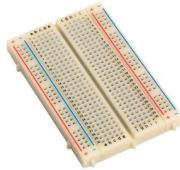
Sensor SHARP: el sensor infraroig que farem servir és un sensor SHARP. Per connectar-lo amb Arduino tenim tres fils: VCC (vermell), GND (negre) i senyal (groc).

Quan detecti un obstacle enviarà una senyal analògica pel fil de senyal. Com més a prop estigui l'objecte, més gran serà la senyal!

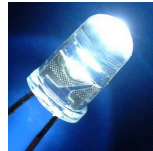
Bones praxis...

La majoria de fabricants proporcionen documentació dels components que venen. És bo llegir-la sempre abans de fer-los servir.

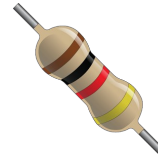
I material d'altres pràctiques anteriors...



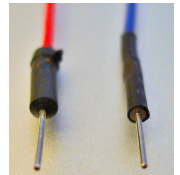
Protoboard



LED



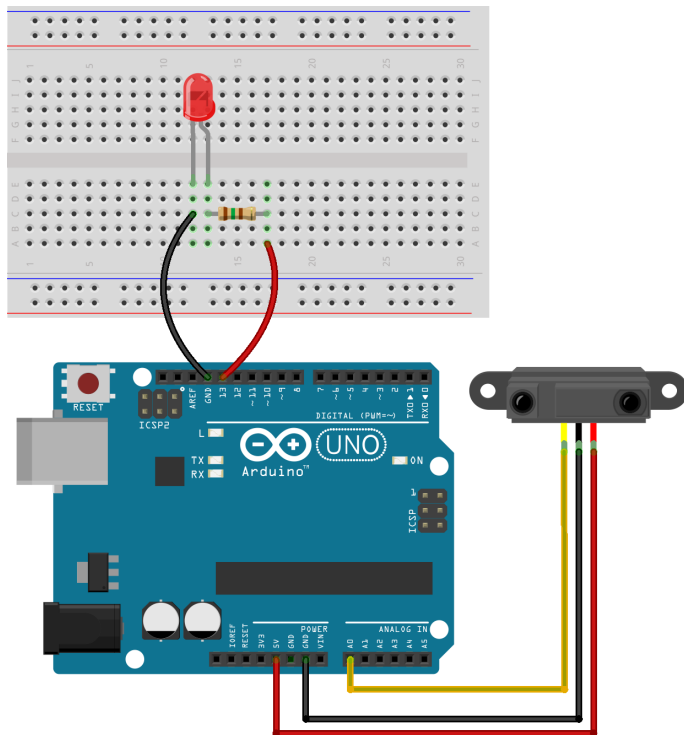
Resistència



Fil
elèctric

El circuit:

Hem de muntar aquest circuit. Atents de no equivocar-nos amb els fils del sensor!



Codi del programa

Aquest codi llegirà el pin analògic del sensor de distància i, en funció de la lectura, encendrà un LED amb més o menys intensitat.

Per llegir el sensor fem servir la funció `analogRead()`, que llegeix un pin analògic i retorna un valor entre 0 i 1024.

Per encendre el LED fem servir la funció `analogWrite()`, que envia un valor entre 0 i 255.

[illegible]

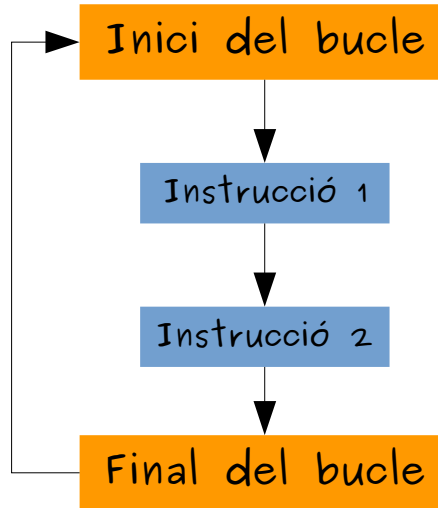
```
    delay(200);  
}
```

Com ho fariem per...

1. ...fer un programa que detecti objectes amb el sensor SHARP i
 - Encengui un LED verd si l'objecte està lluny?
 - Encengui un LED groc si està més a prop?
 - Encengui un LED vermell si està molt a prop?
2. ...modificar el codi i el circuit de la pràctica per encendre un segon LED amb un altre sensor?

Pràctica 6: Bucles FOR i WHILE

Un bucle és una instrucció que ens permet fer que un seguit de línies del programa s'executin indefinidament. Per exemple, la funció `void loop()` és un bucle: quan s'arriba al final es torna al principi, i així indefinidament.



Hi ha dos tipus de bucle: **for** i **while**

Bucles WHILE

```
while(var < 5)
{
    //Fer quelcom
}
```

Els bucles while tenen una condició entre parèntesis. Quan el programa entri en el bucle, i cada cop que arribi al final, mirarà si la condició es compleix (en aquest cas, mirarà si `var` és més petit de 5). Mentre es compleixi, anirà fent les instruccions del bucle.

Bucles FOR

```
for(int i = 0; i < 10; i++)  
{  
    //Fer quelcom  
}
```

Un bucle for s'executarà un nombre predefinit de vegades. A diferència del bucle while, el for no té una sola condició: el que fa és crear una variable que serveixi de comptador i, a cada volta del bucle, suma +1 al comptador. Quan arriba a un cert valor, surt del bucle.

Nota!

Quan es surt del bucle for, la variable i es destrueix.

Això sona complicat, però fixem-nos en el bucle for que hi ha més amunt. Crea una variable i que farà de comptador, i la inicia a 0. Després diu que mentre la i < 10, faci les instruccions del bucle. Cada vegada que s'arribi al final, es tornarà al principi del bucle i es sumarà 1 a la variable i.

En aquest exemple, encendrem un LED mentre no es premi un polsador. L'esquema a muntar serà el mateix que el de la Pràctica 3: PEW! PEW!

Codi del programa

```
#define LED 13
#define BOTO 7

int var; //Aquesta variable guardara l'estat del boto (premut o no)

void setup()
{
    pinMode(LED, OUTPUT);
    pinMode(BOTO, INPUT);
}

void loop()
{
    var = digitalRead(BOTO);

    //Mentre no es premi el boto, repetirem el bucle
    while(var != HIGH)
    {
        digitalWrite(LED, HIGH);
        var = digitalRead(BOTO);
    }
    digitalWrite(LED, LOW);
}
```

Nota!

Un error freqüent és posar un punt i coma (;) al final de les condicions d'un bucle. Fixem-nos que no n'hi ha!

Com ho fariem per...

1. ...modificar la condició del bucle perquè quan el pulsador estigui premut, el led s'encengui?
2. ...fer que el LED s'encengui i s'apagui 3 vegades?
(utilitzant un bucle for, i amb el mateix esquema que la Pràctica 1: BLINK!)

Pràctica 7: Matrius

Hem vist que les variables són com capsetes dins les quals s'hi pot guardar un valor. Què passaria si necessitéssim cent variables? Doncs que les hauríem de declarar una per una:

```
int variable0;  
int variable1;  
int variable2;  
int variable3;  
int variable4;  
int variable5;  
int variable6;  
int variable7;  
int variable8;  
int variable9;  
int variable10;
```

```
int variable11;  
int variable12;  
int variable13;  
int variable14;  
int variable15;  
int variable16;  
int variable17;  
int variable18;  
int variable19;  
int variable20;  
int variable21;  
int variable22;  
int variable23;
```

```
int variable24;
```

```
int variable25;
```

```
int variable26;
```

```
int variable27;
```

```
int variable28;
```

```
...
```

I millor paro. Algú ha arribat fins aquí o m'he quedat sol?

Com podem veure, ens estaríem bastants hores declarant variables!

Si necessitem moltes variables, podem fer servir una matriu. Si una variable era una capseta on hi guardem un valor, una matriu és una calaixera on guardem un valor a cada calaix.

Per declarar una matriu, li hem de dir el tipus de variable que guardarà (int, float, double, char...) i quants espais tindrà aquesta matriu (quin serà el nombre de variables que guardarà).

Per exemple, com seria una matriu de 5 espais?

```
int matriu[5];
```

Equival a una calaixera amb 5 espais on hi podem guardar nombres **enters**...



Nota!

Tant les matemàtiques com la informàtica fan servir matrius. La traducció anglesa de matriu és matrix quan ens referim al camp de les matemàtiques i array quan ens referim a la informàtica.

Si volem accedir a un valor d'una matriu i guardar-lo a una altra variable, podem fer:

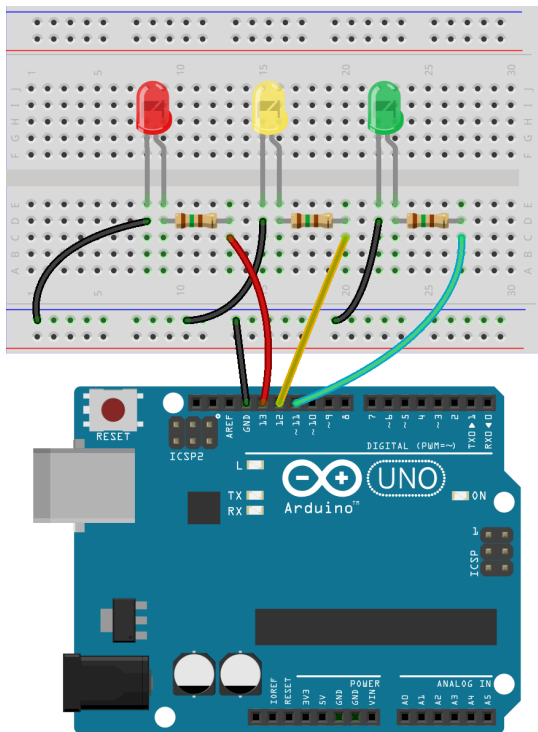
```
var = matriu[3];
```

Això guardarà el quart valor de la matriu dins la variable var. Sí, el quart valor, ho heu llegit bé! Les matrius comencen a partir del 0: l'espai [0] és el primer, [1] és el segon, [2] és el tercer...

Podem crear matrius amb qualsevol nom i tipus, per exemple: `char paraula[4]`, `int nombres[2]`, `double valors[40]`. Si volem, podem emplenar la matriu quan la declarem, i llavors no caldrà posar la longitud entre claudàtors. Exemple: `int valors[] = {1,2,3,4,5,6,7};`

El circuit:

Hem de construir una variant del BLINK! amb tres LEDs.



Codi:

Anem a fer una prova. Crearem una matriu on hi guardarem els valors de 3 pins digitals (que estaran connectats a un LED cada un). Farem servir un bucle for per anar encenent cada un d'aquests pins. Sembla més complicat del que en realitat és.

```
#define NUM_PINS 3

int pins[] = {13,12,11};

void setup()
{
    //En lloc d'anar posant els pins com a OUTPUT un a un,
    //creem un bucle que ens ho faci.
    for(int i = 0; i < NUM_PINS; i++) //'i' prendrà valors entre 0 i 3
    {
        //Per exemple, quan la i valgui 0, la posicio pins[i]
        //sera la posicio pins[0], de manera que aqui establim el pin
        //13 com a OUTPUT (ja que dins de pins[0] hi guardem un 13)
        pinMode(pins[i], OUTPUT);
    }
}

void loop()
{
    //Encenem els LEDs!!!!
    for(int i = 0; i < NUM_PINS; i++)
    {
```

```
digitalWrite(pins[i], HIGH);  
delay(500);  
}  
  
//I els apaguem!!!  
for(int i = 0; i < NUM_PINS; i++)  
{  
    digitalWrite(pins[i], LOW);  
    delay(500);  
}  
}
```

Com ho fariem per...

1. ...canviar els pins dels LEDs? (per exemple, als pins D7, D8, D9)
2. ...afegir un quart LED?
3. ...fer que només s'encengui i s'apagui el LED 13?

Pràctica 8: Funcions

Les funcions, com el seu nom indica, s'encarregaràn de fer una «funció» concreta al nostre programa. De fet, fins ara ja hem utilitzat moltes funcions pròpies d'Arduino (`delay()`, `digitalWrite()`, `analogRead()`...). Però nosaltres també podem crear funcions!

Les funcions poden rebre tot un seguit de paràmetres (anomenats arguments) i poden retornar un valor. Per exemple, suposem que volem crear una funció que ens sumi dos nombres:

```
int suma(int nombre1, int nombre2)
{
    int resultat;
    resultat = nombre1 + nombre2;
    return resultat;
}
```

Aquesta funció rep com a paràmetres dos nombres (`nombre1` i `nombre2`), els suma i torna el resultat. Si ara volem cridar aquesta funció en alguna altra part del codi, haurem d'escriure:

```
valor = suma(5, 2);
```

Ara la variable valor contindrà el resultat de la suma de 5+2.

El tipus de la funció (int, char, float...) ens diu el tipus de valor que retorna. Si la funció no retorna cap valor, serà de tipus void. Per exemple, si volem crear una funció per fer un blink, podem escriure:

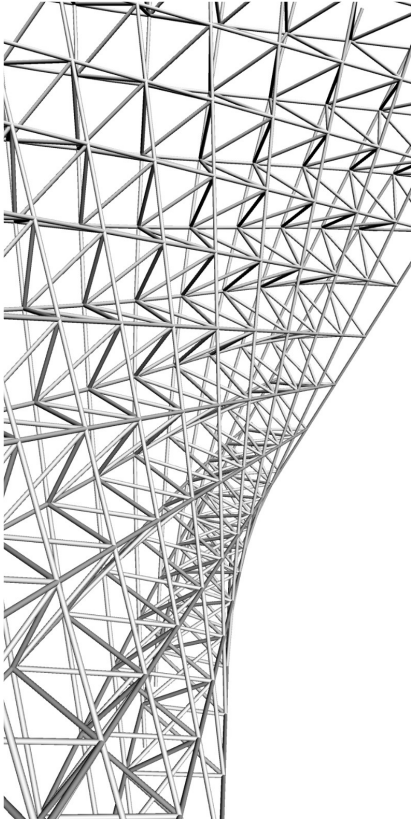
```
void blink(int pin, int temps)
{
    pinMode(pin, OUTPUT);
    digitalWrite(pin, HIGH);
    delay(temps);
    digitalWrite(pin, LOW);
    delay(temps);
}
```

Aquesta funció rep com a paràmetres un pin, un temps i fa un blink. Aquesta funció l'hem d'escriure al final del programa, però la podem cridar a qualsevol punt.

Nota!

També podem crear funcions que no rebin cap paràmetre.

Pràctica EXTRA: Estructures



Fixem-nos amb el nostre DNI. És una targeta on hi ha varis camps:

Nom
Cognoms
Núm de DNI
Data de naixement
...

Si en el teu DNI manca algun d'aquests camps, tens un problema.

En programació un struct (o estructura) és com una targeta amb diferents camps. Per exemple, un struct LED podria tenir dos camps: el pin que té assignat i l'estat (HIGH o LOW) en que està.

En aquesta pràctica crearem un struct LED amb dos camps: pin i estat. Modificarem el contingut d'aquests camps per tal d'encendre i apagar dos LEDs.

Hem de muntar el mateix circuit de la Pràctica 1: BLINK!, però amb la diferència que cal afegir un segon LED al pin D10.

Codi del programa

```
struct LED { //L'estructura LED tindra dos camps
    int estat; //Aqui guardarem si el LED esta a HIGH o LOW
    int pin; //Aqui guardarem el pin del LED
};

//Creem dos LEDs: un es dira primer i l'altre segon
struct LED primer;
struct LED segon;

void setup()
{
    //Emplenem els camps del primer LED
    primer.pin = 13; //El volem al pin 13
    primer.estat = HIGH; //I que comenci estant a HIGH (ences)

    //Emplenem els camps del segon LED
    segon.pin = 10; //El volem connectar al pin 10
    segon.estat = LOW; //I que comenci estant a LOW (apagat)
}

void loop()
{
    digitalWrite(primer.pin, primer.estat);
    digitalWrite(segon.pin, segon.estat);
}
```

Nota!

Dins d'un struct es poden barrejar camps de diversos tipus: int, float, char... fins i tot podem posar-hi matrius!

```
delay(1000);  
primer.estat = LOW;  
segon.estat = HIGH;  
digitalWrite(primer.pin, primer.estat);  
digitalWrite(segou.pin, segon.estat);  
delay(1000);  
primer.estat = HIGH;  
segon.estat = LOW;  
}
```

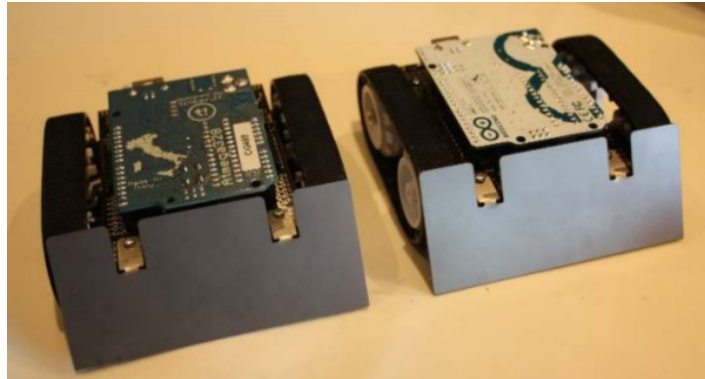
Com ho fariem per...

1. ...afegir un 3r LED?

Muntatge del robot

Ara que ja coneixem Arduino, es hora de programar el nostre robot! Per treballar, farem servir el xassís Zumo, que ja porta gairebé tot el que necessitem: rodes i motors, sensor de reflectància, controlador de motors, etc.

Només hem de col·locar la placa Arduino a sobre, de manera que coincideixin els pins. Arduino serà el cervell del nostre robot: programarem la placa per tal que controli el xassís.



Normalment, hauríem d'escriure les nostres pròpies funcions per moure els motors i controlar els sensors. Per estalviar temps i feina, la llibreria del curset ens dóna ja fetes la majoria d'instruccions, que només haurem de cridar quan les necessitem.

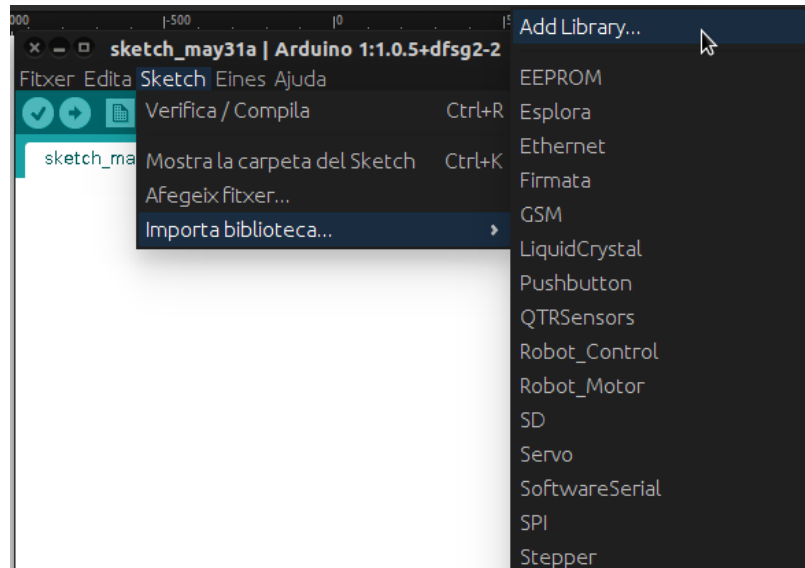
La llibreria del curset es troba a la web d'Astrobanyoles, dins l'apartat de robòtica:

http://www.astrobanyoles.org/robotica/llibreria_curset2015.zip

Haurem de descomprimir-la una vegada descarregada. Sinó, també podem demanar-les als monitors.

Per instal·lar la llibreria un cop descarregada només hem d'obrir la IDE d'Arduino, anar a Sketch

->Import Library->Add Library i buscar l'arxiu.



Part I: el xassís

El nostre robot consta de diverses parts:



Sensor de reflectància: en realitat són sis sensors de color que detecten quanta llum reflecteix un objecte. Quanta més llum reflecteixi, més blanc serà. Això ens servirà per detectar línies blanques i negres al terra i permetrà al robot orientar-se.

Controlador de motors: Arduino no té suficient potència per moure motors. Per això necessitem una altra placa que s'encarregui de controlar-los amb les instruccions que li arriben d'Arduino.



Motors: El xassís porta dos motors a dins. Transformen l'energia elèctrica en moviment.



Suport per a piles Les piles es col·loquen sota el robot. Com es diu al principi d'aquest dossier, a la llista de material, necessitarem 4 piles AA carregades.

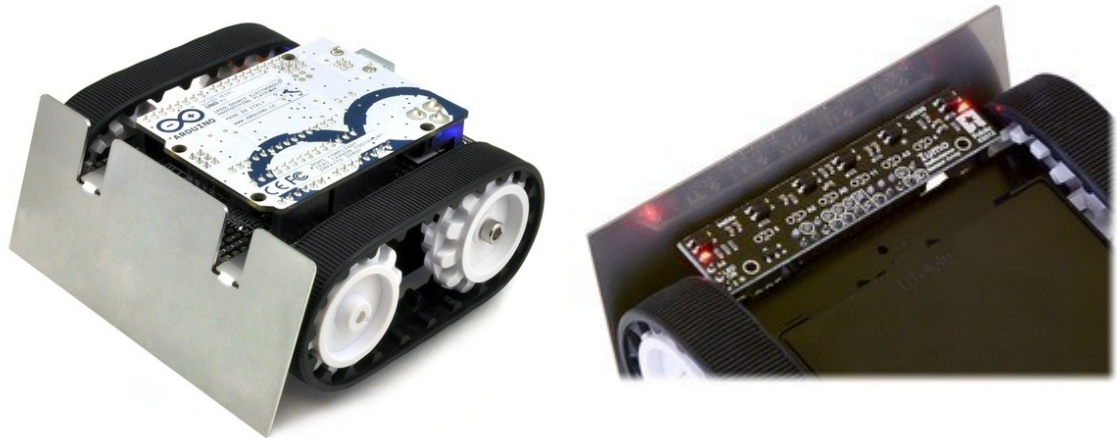
IMPORTANT! No barregem piles de marques diferents ni piles carregades amb piles poc carregades!

Interruptor: engega i apaga el robot. Es troba darrera del xassís.



Pala: no fa res. Eh... però hi queda bé, no?

La placa Arduino s'ha de col·locar damunt del xassís tal i com mostra la imatge de sota a la dreta. Va a pressió, de manera que potser haurem de fer una mica de força. Vigilem que els pins quedin tots al seu lloc! El sensor de reflectància va a sota, mirant cap a fora.



A estones lliures podem dedicar-nos a decorar el xassís! Però sense malmetre el material, (**no feu servir capses ni components dels que us hem donat**) ni feu malbé el xassís. Tampoc el guixeu.

Part II: sobre rodes!

Nota!

A partir d'ara, és molt important portar les piles ben carregades! I esmorzar bé!

És hora de motoritzar-se! En aquesta pràctica aprendrem com moure el robot fent servir les funcions de motors. Només hem de muntar el xassís i connectar l'Arduino a l'ordinador amb un USB.

Comencem per quelcom senzill: farem avançar i retrocedir el robot.

```
#include <Robot.h> //Llibreria a carregar

Robot arduino; //Iniciem l'objecte robot

void setup()
{
    //Res a declarar...
}

void loop()
{
    arduino.motors(100, 100); //Movem els 2 motors a velocitat 100
    delay(1000);
    arduino.motors(-100, -100); //Els fem anar enrere
    delay(1000);
    arduino.parar();
    delay(500);
}
```

En aquest codi tenim algunes instruccions noves:

#include<Robot.h> : aquesta línia importa la llibreria amb totes les funcions per controlar el robot.

Robot arduino : crea un objecte a partir de la classe Robot, que es troba dins la llibreria Robot.h. En aquest curs no entrarem en gaire detalls sobre objectes i classes, només hem de saber que s'ha d'escriure aquesta instrucció al principi del programa.

En cristià, una classe és com un «plànol» del nostre robot. Quan creem l'objecte és com si agaféssim un munt de plastilina i en féssim un objecte seguint les instruccions del plànol. I del nyap que surt, li posem de nom «arduino». Si mai entreu a la universitat per estudiar informàtica, millor no doneu explicacions així als exàmens...

arduino.motors(m1, m2) : canvia les velocitats del motor 1 (m1) i el motor 2 (m2). La velocitat mínima és -400, i la màxima 400. Però millor no farem anar el robot a més de 300.

arduino.parar() : atura els dos motors (els posa a velocitat 0).

Nota!

Per girar el robot, podem fer anar un motor més ràpid que l'altre, o fins i tot, fer-los girar en direccions contràries. Si els motors roden en direccions contràries, el robot girarà sobre el seu propi eix.

Com ho fariem per...

1. ...aturar el robot sense la funció parar()?
2. ...fer girar el robot?
3. ...fer girar el robot sobre si mateix?
4. ...encendre i apagar el LED del pin 13 mentre el robot es mou?

Part III: construcció d'un avoider

Anem a construir el nostre primer robot «seriós»: un avoider! Es tracta d'un robot programat per avançar i esquivar els obstacles que es vagi trobant.

Hem de connectar el sensor SHARP al Pin Analògic 3, i podem enganxar-lo sobre el robot amb una mica de cinta adhesiva de doble cara.

El funcionament del robot serà aquest:

- 0 Mentre no es detecti cap obstacle, el robot avança.
- 1 Si el sensor detecta un obstacle, el robot gira a la dreta.
- 2 Un cop no detecta cap altre objecte, el robot continua avançant.
- 3 per desgràcia l'algorisme no és infal·lible... compte no caigui daltabaix de la taula!

El codi a carregar és aquest. El robot utilitza un bucle while per avançar mentre no detecti cap obstacle, i si el bucle es trenca (és a dir, si es detecta un obstacle) el robot gira. Per tal de llegir el sensor escriurem una funció detectaObstacle() que retorna cert o fals segons si el sensor veu un objecte o no.

```
#include <Robot.h>
#define SENSOR 3

Robot arduino;

void setup()
{
}

void loop()
{
    while(!detectaObstacle()) //Mentre no hi hagi obstacle, ves endavant
    {
        arduino.motors(200,200);
    }
    //Un cop hi hagi un obstacle, el programa surt del bucle

    arduino.motors(200,-200); //Gira a la dreta
}

bool detectaObstacle()
{
```

```
//Si el sensor marca mes de 250, hi ha un objecte
//Per tant, retorna true (cert)
//Si no, retorna false
if(analogRead(SENSOR) > 300)
    return true;

else
    return false;
}
```

Com ho fariem per...

1. ...fer un robot que faci el contrari: giri sobre si mateix però quan detecti un objecte el persegueixi (avanci en línia recta)?
2. ...fer un robot que es mogui fent moviments a l'atzar i es pari quan detecti un objecte. Necessitarem la funció `random(min, max)` que retorna un número a l'atzar entre min i max?

Part IV: seguint línies

Una bona manera de fer anar un robot per allà on volem és fer-lo seguir una línia. Per això farem servir el sensor de reflectància.

Aquesta pràctica consta de dues parts: en la primera aprendrem a llegir el sensor de reflectància amb la funció **llegirSensors()** i enviarem els seus valors per Serial. A la segona part, farem servir la funció **segueixLinia()** per tal de seguir una línia negra al terra.

Nota!

El color mai serà totalment blanc o negre. Per això, caldrà ajustar bé el valor límit que distingeix entre blanc i negre.

Recordem que cal muntar el sensor de reflectància sota el xassís i tenir el robot connectat a l'ordinador en tot moment. Sinó, no podrem llegir els valors!

Codi de la part 1:

```
/*CODI PER LLEGIR SENSORS*/

#include<Robot.h>

Robot arduino;

unsigned int valors[6]; //Guardarem tots els valors dins una matriu
//unsigned int significa que hi guardarem variables de tipus int
//que mai podran ser negatives

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

void loop()
{
    arduino.llegirSensors(valors); //La funcio guarda les lectures dels
    //sensors dins la matriu 'valors'

    //Volem enviar els sis valors per serial. Creem un bucle!
    for(int i = 0; i < 6; i++)
    {
        //Aquestes dues linies escriuran tots els valors en el format:
        // valor1 valor2 valor3 valor4 valor5 valor6
        Serial.print(valors[i]);
```

```
Serial.print(" "); //Deixem un espai entre cada lectura  
}  
Serial.print("\n"); //Quan s'hagin escrit tots, fem un salt de linia  
}
```

En aquest codi utilitzem la funció nova:

arduino.llegirSensors(valors): aquesta funció rep com a paràmetre una matriu de 6 valors de tipus unsigned int i l'emplena amb les lectures dels sensors (que aniran de 0 a 1000).

Codi de la part 2:

```
#include<Robot.h>  
  
Robot arduino;  
  
unsigned int valors[6]; //Aqui hi guardarem les lectures dels sensors
```

```
void setup() {}

void loop()
{
  //La funcio segueixLinia... serveix per seguir linies!
  arduino.segueixLinia(0.25, 0, 0.1, valors);
}
```

Aquesta vegada fem servir una funció nova que és molt important:

arduino.segueixLinia(int Kp, int Ki, int Kd, unsigned int valors): aquesta funció fa servir un algorisme, conegut com a PID controller, per tal de detectar la posició de la línia i corregir l'error movent els motors.

Per tal que el controlador PID corregeixi bé el rumb, se li han de donar tres paràmetres enters que per conveni s'anomenen Kp, Ki i Kd (també li passem com a paràmetre la matriu valors perquè pugui saber on està la línia i calcular l'error).

Per a aquest taller no necessitem saber com funciona per dins aquesta funció. Només hem de saber com es calibra manualment per assaig i error:

-Posem tots els valors a 0:

-Kp = 0

-Ki = 0

-Kd = 0

Nota!

És possible que el codi no funcioni a la primera i que s'hagin de canviar els arguments que es passen a la funció segueixLinia(), perquè aquests seran diferents per a cada robot. De seguida explicarem com es fa.

-Augmentem el valor de K_p fins que veiem que el robot comença a desviar-se bastant de la línia. K_p serà la meitat d'aquest valor.

Exemple: si veiem que el robot es desvia quan $K_p = 0.7$, aleshores posarem $K_p = 0.35$.

- K_i en principi no el necessitem.

-Si veiem que el robot tarda a corregir el seu rumb, podem pujar K_d fins que el robot reaccioni prou ràpid. Un valor de 0.1 o 0.2 hauria de ser suficient.

Com ho fariem per...

1. ...fer un programa que encengui el LED del pin digital 13 si tots els sensors detecten color negre?
2. ...fer un robot que avanci en línia recta fins que tots els sensors detectin negre?

ENHORABONA!!

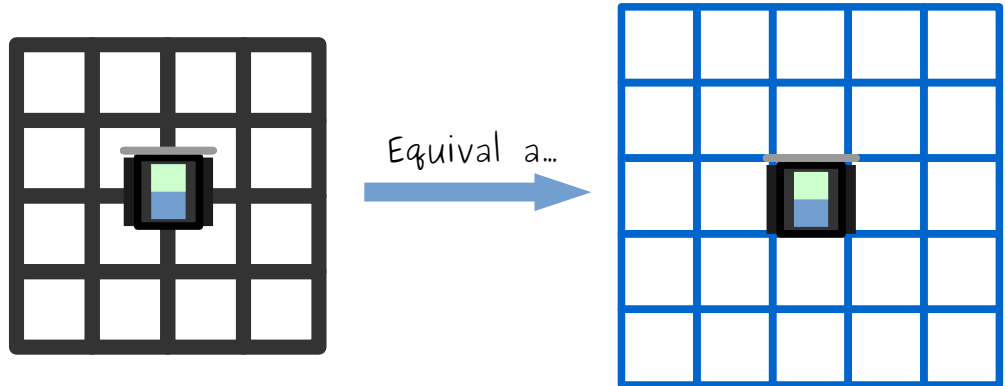
Ja hem acabat totes les pràctiques! Fins aquí arriba la part «guiada» del dossier. A partir d'ara de nosaltres dependrà aplicar tot el que hem après per tal de construir el robot explorador pel nostre compte...



Hey, però tranquils! El dossier ens donarà pistes i ajudes de com ha de ser l'algorisme que hem de construir. L'única cosa que no farà és donar-nos directament el codi. L'hauem de pensar nosaltres!

Com funcionarà el robot explorador?

La idea és construir un robot capaç de trobar el camí més curt cap al seu destí movent-se per un taulell amb una graella. Cada intersecció serà una casella a la que es podrà moure el robot.

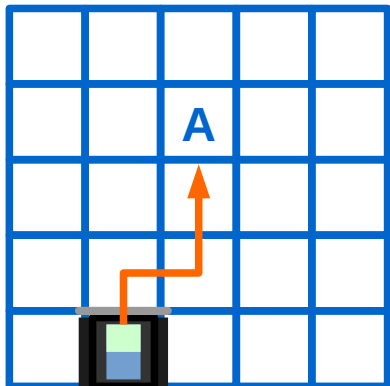


Per fer-ho, tenim dues opcions. La OPCIO 1 està pensada per aquells que els costi més programar o no tinguin tanta experiència. La OPCIO 2 és més complicada, i està dirigida a aquells alumnes que tinguin experiència amb Arduino o se'n surtin molt bé.

OPCIÓ 1:

La primera és programar nosaltres els moviments que haurà de fer el robot (ara tira cap endavant, ara gira, torna a avançar...). Haurem de programar funcions per avançar una casella, girar 90º a la dreta i 90º a l'esquerra, i cridar-les per fer-lo seguir el recorregut.

Per exemple, si volem fer anar el robot de la posició inicial fins al punt A, haurem de programar:



El dia de la demostració, abans de començar, es dirà la distribució dels obstacles.
Haurem de re-programar el robot per arribar al destí sense xocar!

OPCIÓ 2:

Donar les instruccions de moviment a mà està bé. Però no seria millor que fos el mateix robot qui decidís el camí? Quan haguem fet la OPCIÓ 1 i, si sobra temps, a les últimes pàgines s'explica com fer aquest algorisme. És tot un repte!

Guia de l'OPCIÓ 1

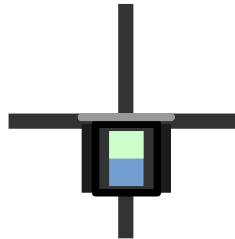
Necessitarem aquestes funcions:

1-Funció endavant()

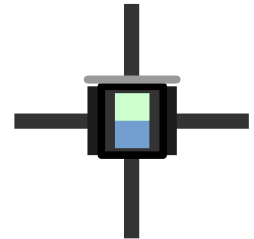
Volem que el robot avanci una casella endavant (recordem que cada intersecció de línies negres serà una casella). Aleshores... com ho hem de fer?



El robot anirà seguint la línia amb la funció segueixLínia();



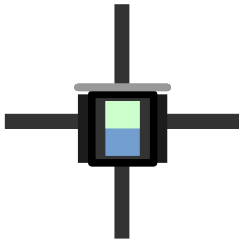
Quan el sensor de reflectància [5] o [0] detecti color negre, significa que ha arribat a una intersecció



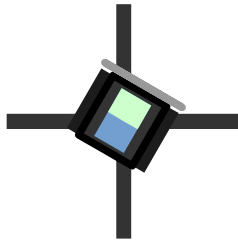
Avançarà una mica més per col·locar-se al centre de la creu i es parará (per exemple, avançarà durant 200ms)

2-Funció giraEsquerra()

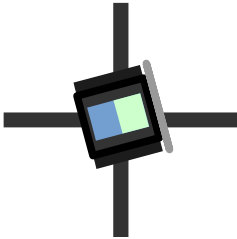
Aquesta funció servirà per fer girar el robot 90º cap a la dreta. Ho farà així:



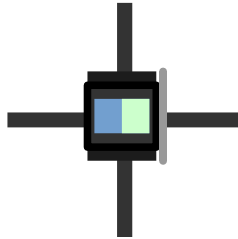
Al principi, el robot té els dos sensors centrals detectant negre



Mentre algun dels dos sensors no detecti color blanc, anirà girant



Una vegada els dos detectin blanc, continuarà girant fins que els dos sensors detectin negre



Es parará i esperarà una altra instrucció

També farà falta una funció `giraEsquerra()`. Només cal copiar la funció `giraDreta()` i canviar la direcció de rotació.

I ja estem a punt! Ara podem cridar aquestes funcions dins el `void setup()` d'Arduino per tal de programar la ruta del nostre robot!

Si ha sobrat temps, us repto a seguir l'Opció 2... i a compartir-la amb els altres companys un cop estigui acabada!

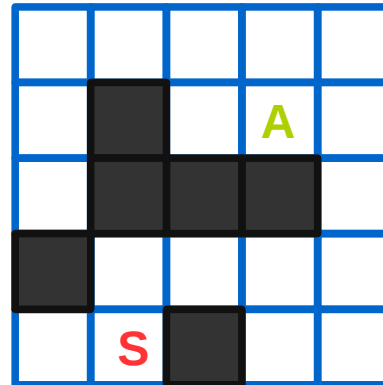
Guia de l'OPCIÓ 2

«Abandon all hope ye who enter here»
-Dante Alighieri, *La Divina Comedia*

El nostre primer pas és trobar un algorisme sobre paper que sigui capaç de donar-nos el camí més curt entre dos punts d'una graella. Imaginem-nos que tenim el següent taulell (on els espais blancs estan buits i els grisos són obstacles). Volem fer sortir el robot del punt S i fer-lo anar al punt A.

Nota!

Abans de fer l'OPCIÓ 2, és millor que intentem fer les funcions de l'OPCIÓ 1. Fins i tot ens poden servir per aquest altre programa!



Bé, potser serà millor que abans de res numerem les caselles. Volem anar de B5 (la casella S) a D2 (la casella A)

	A	B	C	D	E
1					
2				A	
3					
4					
5		S			

Per començar, creem una llista. En aquesta llista hi anirem guardant totes les coordenades, però d'una manera especial. Per cada casella guardarem la seva posició (lletra i número) i també un tercer valor que anomenarem «cost». Al principi, a la nostra llista només hi haurà la casella FINAL (A) amb cost = 0. Sí, la casella final, no és un error.

	A	B	C	D	E
1					
2				A	
3					
4					
5		S			

Coordenades

Cost

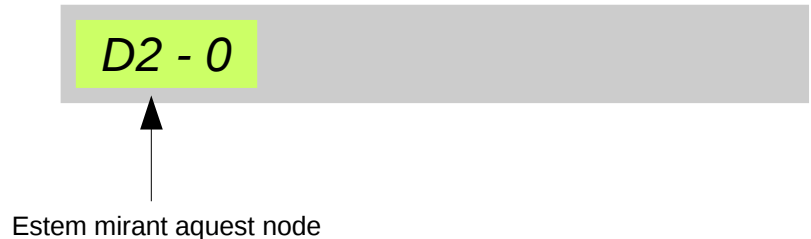
Llista

D2 - 0

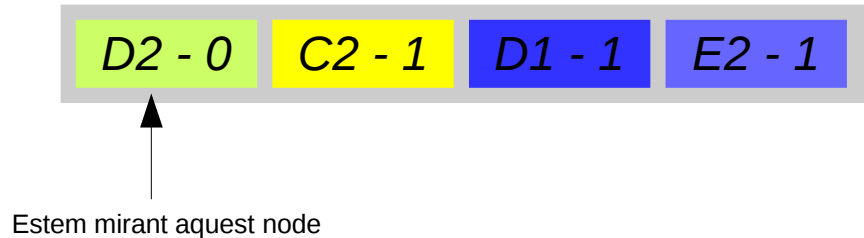
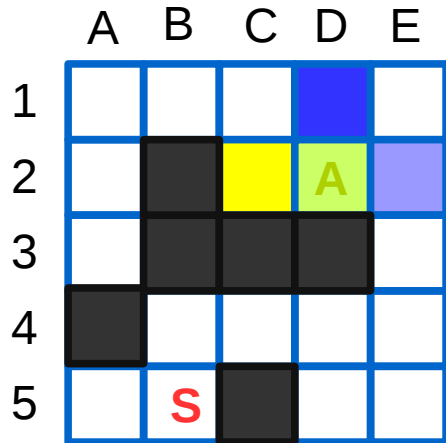
Ara el que farem és anar mirant tots els elements de la llista, un a un.

Per a cada element, ens fixem en les quatre caselles que té adjacents. El primer element de la llista és D2, i té com a caselles adjacents D1, C2, D3, E2.

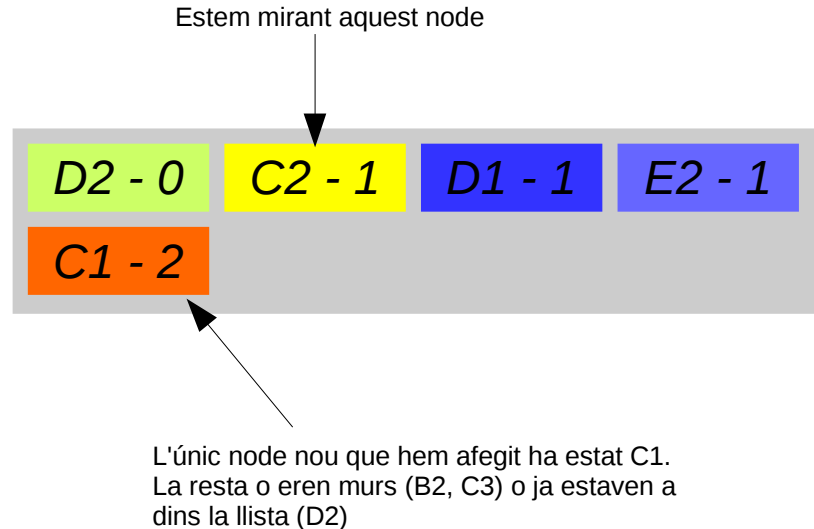
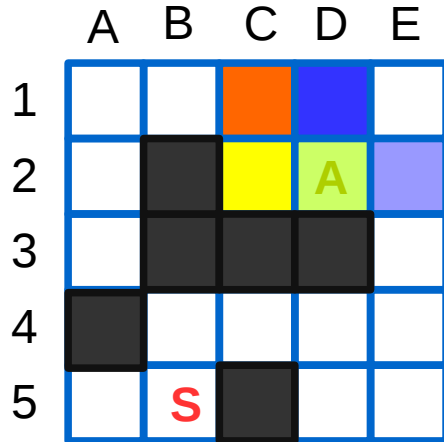
	A	B	C	D	E
1					
2				A	
3					
4					
5		S			



Totes les caselles que no siguin murs i no estiguin ja a la llista les afegim sumant 1 al comptador del node que estem mirant.

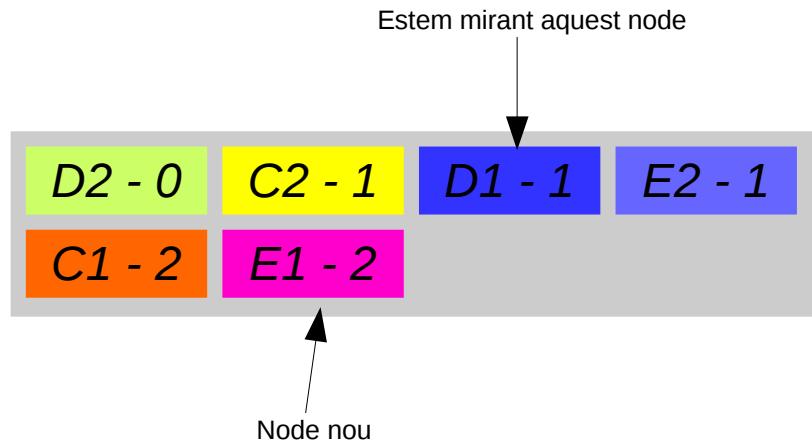


Mirem el següent node de la llista (C2, color groc) i tornem a repetir el procés. De les quatre caselles que té adjacents, afegim a la llista els que no estiguin ja a la llista i no siguin murs. El seu comptador serà igual al comptador de la casella que estem mirant +1.



Repetim el procés amb el següent node de la llista (D1)

	A	B	C	D	E
1					
2				A	
3					
4					
5		S			



Anem repetint el procés. Tard o d'hora, arribarem a la **casella S**.

Quan posem la **casella S** dins la llista, parem. Ja haurem trobat una ruta.

	A	B	C	D	E
1					
2				A	
3					
4					
5		S			

D2 - 0	C2 - 1	D1 - 1	E2 - 1
C1 - 2	E1 - 2	E3 - 2	B1 - 3
E4 - 3	A1 - 4	D4 - 4	E5 - 4
A2 - 5	C4 - 5	D5 - 5	A3 - 6
B4 - 6	B5 - 7		

Node final (S)

Tenim una ruta des del node de sortida (S) i el node d'arribada (A). Col·loquem tots els costos sobre les caselles corresponents. Les caselles que no estiguin a la llista, tindràn un pes de 99 (o qualsevol altre nombre gran)

	A	B	C	D	E
1	4	3	2	1	2
2	5	99	1	0	1
3	6	99	99	99	2
4	99	6	5	4	3
5	99	7	99	5	4

D2 - 0	C2 - 1	D1 - 1	E2 - 1
C1 - 2	E1 - 2	E3 - 2	B1 - 3
E4 - 3	A1 - 4	D4 - 4	E5 - 4
A2 - 5	C4 - 5	D5 - 5	A3 - 6
B4 - 6	B5 - 7		

El robot comença a la casella 7. Per trobar la ruta fins al punt 0, es mou a la casella adjacent que tingui el cost més baix:

	A	B	C	D	E
1	4	3	2	1	2
2	5	99	1	0	1
3	6	99	99	99	2
4	99	6	5	4	3
5	99	7	99	5	4

El robot comença aquí

D2 - 0	C2 - 1	D1 - 1	E2 - 1
C1 - 2	E1 - 2	E3 - 2	B1 - 3
E4 - 3	A1 - 4	D4 - 4	E5 - 4
A2 - 5	C4 - 5	D5 - 5	A3 - 6
B4 - 6	B5 - 7		

Com ho implementem? Aquest dossier proposa una forma de fer-ho, però de ben segur que n'hi ha moltes altres. I fins i tot més eficients!!!

0 En primer lloc, hem de donar dues matrius al robot.

La primera serà una matriu de dues dimensions que contindrà el mapa. Per exemple, suposant que el taulell sigui de 6x8, podem fer:

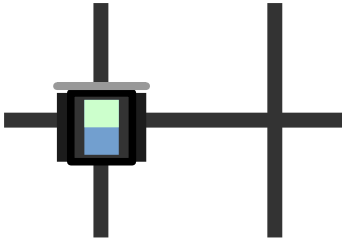
```
int mapa[MAPAX][MAPAY] = {  
{0,0,0,0,0,0,0,0},  
{0,7,7,0,7,7,0,0},  
{0,0,0,7,0,0,0,0},  
{7,0,7,0,0,7,7,0},  
{2,0,7,0,7,0,0,7},  
{0,0,7,0,0,0,0,1}  
};
```

On els 0 són espais buits, els 7 obstacles, 1 el punt de sortida i 2 el d'arribada. La segona matriu serà igual de gran, però buida.

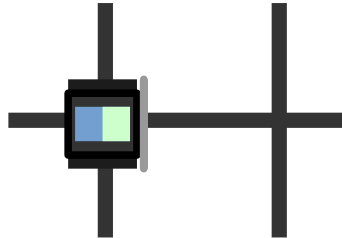
L'emplenarem al final amb els costos.

Per emplenar podem triar qualsevol número.

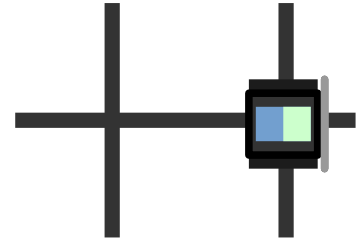
1 Necessitem vàries funcions: una que faci moure el robot una casella cap amunt, una altra que mogui el robot una casella cap avall, una altra a la dreta i una altra cap a l'esquerra. Per fer-ho, haurem de guardar l'orientació del robot en una variable i cridar les funcions que hem construït a l'OPCIÓ 1:



Volem moure el robot una casella a la dreta. La seva orientació és cap a **DALT**, per tant cal cridar la funció giraDreta() una vegada i la funció endavant()



Hem cridat la funció giraDreta(). Ara el robot ha canviat l'orientació, per tant canviem la variable a **DRETA**.



Quan el robot ja té l'orientació que volem, cridem la funció endavant()

2 Un struct casella amb tres camps: posicioX, posicioY i cost.

Després creem dos nodes (que anomenarem inicial i final) i una matriu d'una dimensió que serà la nostra llista.

```
struct casella {  
    int x;  
    int y;  
    int cost;  
};  
struct casella llista[MAPAX*MAPAY];  
struct casella final;  
struct casella inicial;
```

Els nodes inicial i final seran l'inici del robot i el destí en el mapa.

3 Una funció inicialitza() que vagi recorrent tot el mapa amb un bucle i guardi les coordenades de la posició inicial i final del mapa a les caselles inicial i final. L'estructura de la funció seria:

```
Funcio inicialitza():  
    for(int i = 0; i < MAPAX; i++):  
        for(int j = 0; j < MAPAY; j++):  
            Si mapa[i][j] == sortida:  
                inicial.x = i  
                inicial.y = j  
                inicial.pes = 0  
            Sino si mapa[i][j] == arribada:  
                final.x = i  
                final.y = j  
                final.pes = 0
```

Cal traduir aquest codi al codi d'Arduino.

4 Una funció que donada una casella, retorni cert o fals si està o no a la llista:

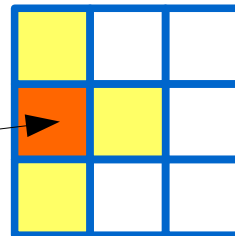
```
bool repetit(struct node entrada):  
    for(int i = 0; i < n; i++):  
        Si llista[i].x == entrada.x and llista[i].y  
== entrada.y):  
            return true  
        return false
```

5 Una funció que calculi la ruta a seguir. En primer lloc, cridarà la funció inicialitza(). Posarà a la posició 0 de la llista la casella final.

Després anirà iterant per totes les caselles de la llista i l'anirà emplenant d'acord amb les normes que hem explicat al principi (afegeix les caselles adjacents que no siguin obstacles ni estiguin ja a la llista).

Ep, però hem d'anar en compte de no sortir-nos de la matriu! Els nodes dels costats tenen només 2 o 3 caselles adjacents:

Aquesta casella només en té 3 d'adjacents.



També haurem de saber quants elements hi ha dins la llista. Podem guardar-los dins una variable 'n'. Cada vegada que afegim una casella, sumarem 1 a 'n'.

```

funcio calculaRuta():
    inicialitza();
    struct casella provisional
    llista[0].x = final.x
    llista[0].y = final.y
    llista[0].cost = final.cost

    for(int i = 0; i < n; i++):
        Si llista[i].x == inicial.x and llista[i].y
== inicial.y:
            inicial.cost = llista[i].cost +1
            Surt del bucle
        Si llista[i].x > 0:
            provisional.x = llista[i].x -1
            provisional.y = llista[i].y
            provisional.cost = llista[i].cost +1
            Si !repetit(provisional) and
mapa[provisional.x][provisional.y] != obstacle:
                llista[n].x = provisional.x
                llista[n].y = provisional.y
                llista[n].cost = provisional.cost
                n = n+1

```

```

        Si llista[i].x < MAPAX-1:
            provisional.x = llista[i].x+1
            provisional.y = llista[i].y
            provisional.cost = queue[i].cost +1
            Si !repetit(provisional) and
mapa[provisional.x][provisional.y] != obstacle:
                llista[n].x = provisional.x
                llista[n].y = provisional.y
                llista[n].cost = provisional.cost
                n = n+1
        Si llista[i].y > 0:
            provisional.x = llista[i].x
            provisional.y = llista[i].y-1
            provisional.cost = llista[i].cost +1
            Si !repetit(provisional) and
mapa[provisional.x][provisional.y] != obstacle:
                llista[n].x = provisional.x
                llista[n].y = provisional.y
                llista[n].cost = provisional.cost
                n = n+1

```

```

Si llista[i].y < MAPAY-1:
    provisional.x = llista[i].x
    provisional.y = llista[i].y+1
    provisional.cost = llista[i].cost +1
    Si !repetit(provisional) and
mapa[provisional.x][provisional.y] != obstacle:
    llista[n].x = provisional.x
    llista[n].y = provisional.y
    llista[n].cost = provisional.cost
    n = n+1

```

6 Una funció que empleni el segon mapa de costos amb nombres elevats i, després, empleni les caselles que estiguin a la llista amb els costos corresponents.

```

Funcio emplenar():
    for(int i = 0; i < MAPAX; i++):
        for(int j = 0; j < MAPAY; j++):
            mapa_costos[i][j] = 99
    for(int i = 0; i < n; i++):
        mapa_costos[llista[i].x][llista[i].y] =
llista[i].cost
    }

```

7 Per acabar, la funció `seguirRuta()`. Ja tenim la matriu de pesos plena, ara només es tracta de fer que el robot es mogui a la casella adjacent amb el cost més baix fins que no arribi a la casella final! L'estructura a seguir serà:

Guardar la casella inicial del robot

Mirar les quatre caselles que té adjacents:

- Si la casella de dalt té el cost més baix, el robot es mou cap amunt.
- Si la casella de la dreta té el cost més baix, el robot es mou cap a la dreta.
- Si la casella de l'esquerra té el cost més baix, el robot es mou cap a la esquerra.
- Si la casella de baix és la que té el cost més baix, el robot es mou cap abaix.
- Actualitzem la posició del robot (canviant les coordenades de la casella inicial per les de la nova casella).

-Repetim aquest procés fins que la casella a la que estigui el robot sigui la casella final.

I això és tot! Ara podem anar a la funció `void setup()` i cridar les funcions que hem fet:

```
void setup()
{
    calculaRuta();
    emplenar();
    seguirRuta();
}
```

ANEXE: Sobre Mart

Si hem de viatjar a Mart per rescatar el robot i el seu carregament, el millor serà tenir una idea de com és el planeta...

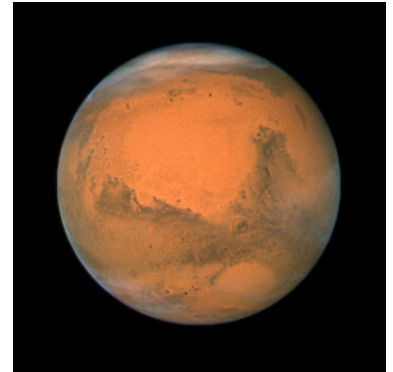
Mart:

És el quart planeta del Sistema Solar, també conegut com el planeta vermell a causa al seu color. Això va fer que els antics romans el relacionessin amb la sang i per això li van posar el nom del Déu de la Guerra: Mart.

Mart té una atmosfera molt dèbil, formada per diòxid de carboni. Conté aproximadament un 0.03% d'aigua.

Els estudis demostren que Mart, en un passat llunyà, va tenir una atmosfera semblant a la de la Terra: més densa, amb núvols, precipitacions, rius i oceans. Sobre la seva superfície desèrtica encara s'hi veuen antigues illes, costes i rius.

Les grans diferències de temperatura a la superfície de mart provoquen grans vents i tempestes de sorra.



Valles Marineris:

És un gegantí sistema de congostos que travessa l'equador de Mart, situat a l'est de la regió de Tharsis, amb 4.500 km de longitud, 200 km d'amplària i 11 km de profunditat màxima.

Alba Patera:

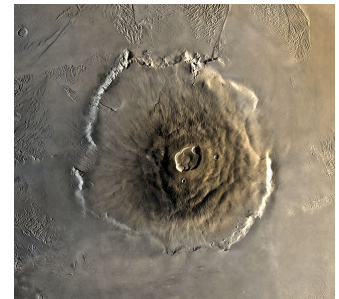
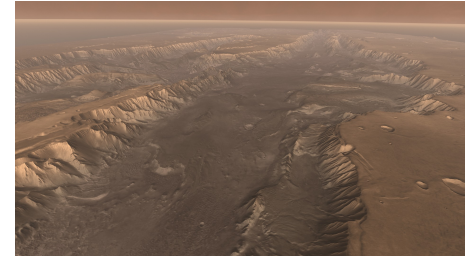
Volcà situat al nord de la regió de Tharsis, amb una gran base d'uns 1.600 km de diàmetre i només 6 km d'altura. És el volcà més gran del Sistema Solar, quant a àrea i volum.

Tharsis:

Enorme planúria volcànica situada a la zona equatorial de Mart.

Mont Olimp de Mart:

És el major volcà del sistema solar. Situat a la regió de Tharsis, té 23 quilòmetres d'altura, aproximadament tres vegades l'alçada de l'Everest. No és recomanable anar-hi a fer excursionisme.



Cràter Everswalde:

És un cràter d'aproximadament 800 metres de profunditat que, en un passat, podria haver estat un llac. Els seus sediments van ser analitzats per la Mars Express.

Hellas Planitia:

És una regió plana i circular situada a l'hemisferi sud de Mart. Té un diàmetre de 2.300 quilòmetres i es sospita que va ser el lloc d'impacte d'un meteorit fa 4 milions d'anys.

Cràter Asimov:

És un enorme cràter de 84 quilòmetres de diàmetre que deu el seu nom a l'escriptor i bioquímic nord-americà Isaac Asimov.



Terra Sirenum:

Es tracta d'una enorme regió de la superfície marciana, localitzada a l'hemisferi meridional del planeta. És una zona irregular i muntanyenca, a causa dels seus nombrosos cràters. Entre ells s'inclou el cràter de Newton.

