

# Curs d'iniciació a Arduino



# Índex de continguts

1 - ¿Què és i quines parts té arduino?.....	4
1.1 - Objectius.....	4
1.2 - ¿Què és arduino?.....	4
1.3 - Parts d'un arduino.....	4
1.3.1 - Entrades.....	4
1.3.2 - Eixides.....	5
1.3.3 - Altres pins.....	5
1.3.4 - Alimentació.....	5
1.3.5 - Comunicació.....	5
1.3.6 - Shields.....	5
1.4 - ¿Com funciona arduino?.....	6
2 - L'entorn de desenvolupament integrat Arduino (IDE).....	7
2.1 - Objectius.....	7
2.2 - Introducció.....	7
2.3 - Escriure esbossos (sketch).....	7
2.4 - Els botons de la barra d'eines.....	8
2.5 - Fitxer.....	8
2.6 - Edita.....	9
2.7 - Esbós.....	9
2.8 - Eines.....	10
2.9 - Ajuda.....	10
3 - Estructura d'un programa de arduino i ordres bàsiques.....	11
3.1 - Objectius.....	11
3.2 - Estructura d'un programa Arduino.....	11
3.3 - Primeres instruccions en Arduino C++.....	11
3.4 - Resum.....	14
4 - El nostre primer circuit.....	15
4.1 - Objectius.....	15
4.2 - Material requerit.....	15
4.2.1 - Algunes idees bàsiques sobre electrònica.....	15
4.3 - El nostre primer circuit electrònic.....	16
4.4 - Resum.....	18
5 - Les entrades digitals.....	19
5.1 - Objectius.....	19
5.2 - Material requerit.....	19
5.3 - Entrades digitals.....	19
5.4 - Esquema electrònic del circuit.....	19
5.5 - Llegint els polsadors.....	21
5.6 - Resum.....	22
6 - Muntatge pull-up i pull-down.....	23
6.1 - Objectius.....	23
6.2 - Introducció.....	23
6.3 - Pull-up.....	23
6.4 - Pull-down.....	24
6.5 - Resum.....	24
7 - Les entrades analògiques.....	25
7.1 - Objectius.....	25
7.2 - Material requerit.....	25
7.3 - Els potenciòmetres.....	25
7.4 - Circuit per a protoboard.....	26
7.5 - Arduino i les entrades analògiques.....	27
7.6 - Usant les portes analògiques.....	28
7.7 - Un últim comentari.....	29
7.8 - Resum.....	30
8 - Les eixides analògiques.....	31
8.1 - Objectius.....	31
8.2 - Material requerit.....	31
8.3 - Analògic i digital.....	31
8.4 - Eixides quasi analògiques.....	32
8.5 - Modificant la lluentor d'un LED.....	33
8.6 - Resum.....	34
9 - Comunicació amb l'exterior.....	36
9.1 - Objectius.....	36
9.2 - Material requerit.....	36
9.3 - Comunicació Sèrie amb el món exterior.....	36
9.4 - Establint la comunicació Sèrie.....	37
9.5 - Rebut missatges a través del port Sèrie.....	40
9.6 - Resum.....	41
10 - Crear gràfiques utilitzant el port sèrie.....	43
10.1 - Objectius.....	43
10.2 - Material requerit.....	43
10.3 - Dibuixar una gràfica utilitzant el serial traçador.....	43

10.4 - Com incloure més variables.....	44
Resum.....	45
11 - Annex 1. Alimentació de Arduino.....	46
11.1 - Mecanismes d'alimentació per al arduino.....	46
11.2 - Alimentar el arduino mitjançant USB.....	47
11.3 - El jack d'alimentació externa del arduino.....	47
11.3.1 - Limitants del regulador de voltatge (sobrecalfament del arduino).....	48
11.4 - Alimentar el Arduino a través del pin Vin.....	48
11.5 - Alimentar el Arduino a través del pin 5V.....	49

# 1 - ¿Què és i quines parts té arduino?

## 1.1 - Objectius

- Presentar la placa arduino
- Saber cóm alimentar-la
- Diferenciar les distintes parts funcionals
- Saber cóm funciona arduino

## 1.2 - ¿Què és arduino?

Arduino és una plataforma de prototips electrònica de codi obert (open-source) basada en una senzilla placa amb entrades i eixides, en un entorn de desenvolupament que està basat en el llenguatge de programació Processing. És un dispositiu que connecta el món físic amb el món virtual, o el món analògic amb el digital.

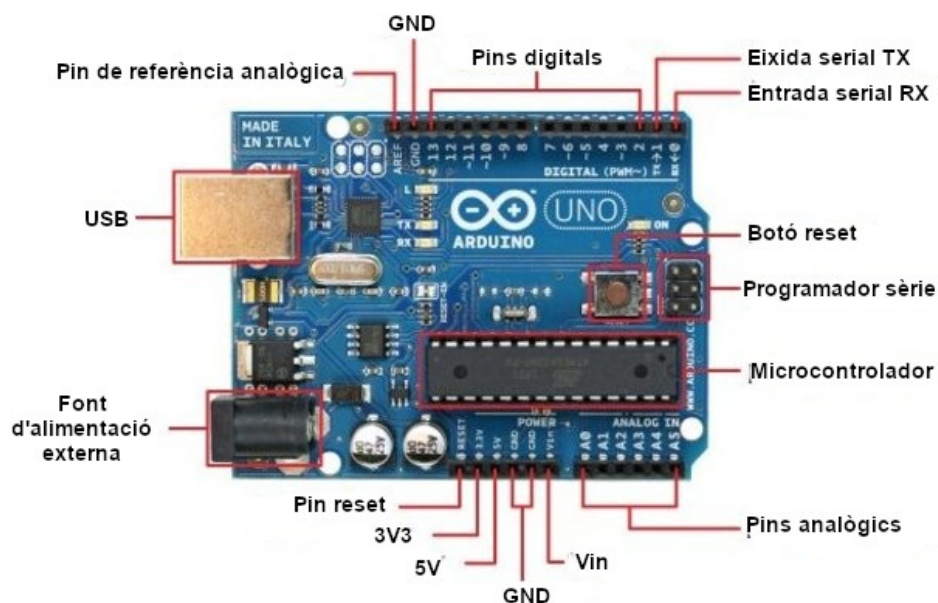


Figura 1: ¿Què és i quines parts té arduino? - Pins de Arduino

## 1.3 - Parts d'un arduino

El arduino com tot component aquesta conformat diferents parts com són entrades, eixides, alimentació, comunicació i shields.

### 1.3.1 - Entrades

Són els pins de la nostra placa que podem utilitzar per a fer lectures. En la placa «Uno» són els pins digitals (del 0 al 13) i els analògics (de l'A0 a l'A5).

### 1.3.2 - Eixides

Els pins d'eixides s'utilitzen per a l'enviament de senyals. En aquest cas els pins d'eixida són només els digitals (0 a 13).

### 1.3.3 - Altres pins

També tenim altres pins com els **GND** (terra), **5V** que proporciona 5 Volts, **3.3V** que proporciona 3.3 Volts, els pins **REF** de referència de voltatge, **TX** (transmissió) i **RX** (lectura) també usats per a comunicació serial, **RESET** per a reinicialitzar, **Vin** per a alimentar la placa i els pins **ICSP** per a comunicació **SPI**.

### 1.3.4 - Alimentació

Com hem vist el pin **Vin** serveix per a alimentar la placa però el més normal és alimentar-lo pel jack d'alimentació usant una tensió de 7 a 12 Volts. També podem alimentar-ho pel port USB però en la majoria d'aplicacions no el tindrem connectat a un ordinador. (Veure annex 1 per a més informació).

### 1.3.5 - Comunicació

En els nostres tutorials ens comunicarem amb Arduino mitjançant USB per a carregar els programes o enviar/rebre dades. No obstant això no és l'única forma que té Arduino de comunicar-se. Quan inserim una shield aquesta es comunica amb la nostra placa utilitzant els pins ICSP (comunicació ISP), els pins 10 a 13 (també usats per a comunicació ISP), els pins TX/RX o qualsevol dels digitals ja que són capaços de configurar-se com a pins d'entrada o eixida i rebre o enviar polsos digitals.

### 1.3.6 - Shields

Es diu així a les plaques que s'insereixen sobre Arduino a manera d'escut ampliant les seues possibilitats d'ús. En el mercat existeixen infinitat de shields per a cada tipus de Arduino. Algunes de les més comunes són les de Ethernet, Wi-Fi, Ultrasons, Pantalles LCD, relés, matrius LED's, GPS.

Arduino està constituït en el maquinari per un micro controlador principal anomenat Atmel AVR de 8 bits (que és programable amb un llenguatge d'alt nivell), present en la majoria dels models de arduino.

Característiques generals de totes les plaques arduino

- El microprocessador ATmega328
- Memòria Flash 32 KB (2 KB per al bootloader)
- SRAM 1 KB
- EEPROM 512 byte
- Microcontrolador ATmega328
- Completament autònom: Una vegada programat no necessita estar connectat al PC
- Velocitat de rellotge 16 MHz
- 13 pins per a entrades/eixides digitals (programables)
- 5 pins per a entrades analògiques
- 6 pins per a eixides analògiques (eixides PWM)

- Voltatge d'operació 5V
- Voltatge d'entrada (recomanat) 7-12 V
- Voltatge d'entrada (límits) 6-20 V
- DC corrent I/O Pin 40 mA
- DC corrent 3.3V Pin 50 mA

## 1.4 - ¿Com funciona arduino?

El arduino és una placa basada en un microcontrolador, específicament un ATMEL. Un microcontrolador és un circuit integrat (podríem parlar d'un microhip) en el qual es poden gravar instruccions. Aquestes instruccions s'escriuen utilitzant un llenguatge de programació que permet a l'usuari crear programes que interactuen amb circuits electrònics.

Normalment un microcontrolador posseeix entrades i eixides digitals, entrades i eixides analògiques i entrades i eixides per a protocols de comunicació. Un arduino és una placa que compta amb tots els elements necessaris per a connectar perifèrics a les entrades i eixides del microcontrolador. Es tracta d'una placa impresa amb tots els components necessaris per al funcionament del micro i la seua comunicació amb una computadora a través de comunicació serial.

La comunicació serial és un protocol de comunicació que alguna vegada va ser molt utilitzat a través dels ports serie que portaven les computadores d'antany.

## 2 - L'entorn de desenvolupament integrat Arduino (IDE)

### 2.1 - Objectius

- Presentar l'entorn de desenvolupament integrat
- Diferenciar les seues parts
- Conèixer les funcions

### 2.2 - Introducció

L'entorn de desenvolupament integrat Arduino (IDE) és una aplicació multiplataforma que es connecta a les plaques Arduino per carregar programes i comunicar-se amb ells.

L'entorn de desenvolupament integrat Arduino (IDE) conté:

1. un editor de text per escriure codi,
2. una àrea de missatges,
3. una consola de text,
4. una barra d'eines amb botons per a funcions habituals
5. una sèrie de menús.

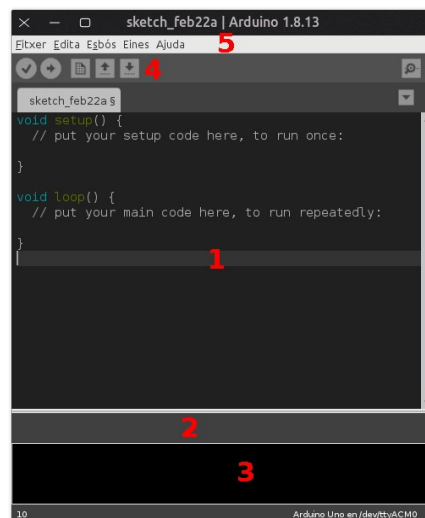


Figura 2: L'entorn de desenvolupament integrat Arduino (IDE). Parts de l'IDE arduino

### 2.3 - Escriure esbossos (sketch)

Els programes escrits amb el programari Arduino (IDE) s'anomenen esbossos o «sketches».

Aquests esbossos s'escriuen a l'editor de text i es guarden amb l'extensió de fitxer **.ino**. L'editor té funcions per copiar / enganxar i per cercar / substituir text.

L'àrea de missatges proporciona comentaris mentre es desa i s'exporta i també mostra errors. La

consola mostra la sortida de text del programari Arduino (IDE), inclosos missatges d'error complets i altra informació.

A l'extrem inferior dret de la finestra es mostra la placa configurada i el port sèrie.

## 2.4 - Els botons de la barra d'eines

Els botons de la barra d'eines us permeten verificar i penjar programes, crear, obrir i desar esbossos i obrir el monitor sèrie.



Figura 3: L'entorn de desenvolupament integrat Arduino (IDE). La barra d'eines

- **Verifica.** Comprova el codi per si hi han errors abans de compilar-lo.
- **Puja.** Compila el codi i el penja a la placa configurada.
- **Nou.** Crea un esbós nou.
- **Obre.** Presenta un menú amb tots els esbossos del sketchbook. Si feu clic a un, s'obrirà en una nova finestra.
- **Desa.** Guarda el vostre esbós.
- **Monitor sèrie.** Obre el monitor sèrie.

Es troben ordres addicionals als cinc menús: **Fitxer**, **Edita**, **Croquis**, **Eines**, **Ajuda**. Els menús són sensibles al context, cosa que significa que només hi ha disponibles els elements rellevants per al treball que s'està realitzant actualment.

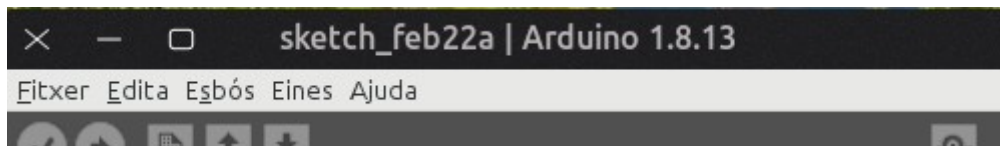


Figura 4: L'entorn de desenvolupament integrat Arduino (IDE). La barra de menús

## 2.5 - Fitxer

- **Nou.** Obre una nova finestra de l'editor, amb l'estructura mínima d'un esbós ja instal·lada.
- **Obre.** Permet carregar un fitxer d'esbós navegant per les unitats i carpetes de l'ordinador.
- **Obre Recent.** Ofereix una llista breu dels esbossos més recents, a punt per obrir-se.
- **Sketchbook.** Mostra els esbossos actuals dins de l'estructura del Sketchbook; en fer clic a qualsevol nom s'obrirà l'esbós corresponent en una nova finestra de l'editor.
- **Exemples.** Qualsevol exemple proporcionat pel programari Arduino (IDE) o la biblioteca apareix en aquest element del menú. Tots els exemples s'estructuren en un arbre que permet un fàcil accés per tema o biblioteca.
- **Tanca.** Tanca la finestra del programari Arduino des del qual es fa clic.
- **Desa.** Guarda l'esbós amb el nom actual. Si el fitxer no s'ha nomenat abans, es proporcionarà un nom a la finestra "Anomena i desa ...".
- **Anomena i desa ...** Permet desar l'esbós actual amb un nom diferent.



- **Configuració de la pàgina.** Mostra la finestra de configuració de la pàgina per imprimir.
- **Imprimeix.** Envia l'esbós actual a la impressora segons els paràmetres definits a Configuració de pàgina.
- **Preferències.** Obre la finestra Preferències, on es poden personalitzar alguns paràmetres de l'IDE, com a idioma de la interfície IDE.
- **Surt.** Tanca totes les finestres IDE. Els mateixos esbossos oberts quan es va triar **Surt** es tornaran a obrir automàticament la propera vegada que iniciu l'IDE.

## 2.6 - Edita

- **Desfés / Refés.** Retrocedeix un o més passos que heu fet mentre editeu; quan torneu enrere, podeu avançar amb Refés.
- **Retalla.** Elimina el text seleccionat de l'editor i el col·loca al porta-retalls.
- **Copia.** Duplica el text seleccionat a l'editor i el col·loca al porta-retalls.
- **Copia per al fòrum.** Copia el codi del vostre esbós al porta-retalls en un formulari adequat per publicar-lo al fòrum, amb color de sintaxi.
- **Copia com a HTML.** Copia el codi del vostre esbós al porta-retalls com a HTML, adequat per inserir-lo a pàgines web.
- **Enganxa.** Posa el contingut del porta-retalls a la posició del cursor, a l'editor.
- **Selecciona-ho tot.** Selecciona i resalta tot el contingut de l'editor.
- **Ves a la línia ...** Demana i posiciona el cursor a la línia indicada.
- **Comenta o descomenta.** Posa o suprimeix el marcador de comentaris // al començament de cada línia seleccionada.
- **Augmenta / Disminueix el sagnat.** Afegeix o resta un espai al principi de cada línia seleccionada, movent el text un espai a la dreta o eliminant un espai al principi.
- **Incrementa / Redueix la mida del tipus de lletra.** Fa més gran / menuda la mida de la lletra del sketch
- **Cerca.** Obre la finestra *Cerca i reemplaça* on podeu especificar text per buscar dins del sketch actual segons diverses opcions.
- **Cerca el següent.** Destaca la següent ocurrència (si n'hi ha) de la cadena especificada com a element de cerca a la finestra *Cerca*, en relació amb la posició del cursor.
- **Cerca l'anterior.** Destaca l'ocurrència anterior (si n'hi ha) de la cadena especificada com a element de cerca a la finestra *Cerca* en relació amb la posició del cursor.

## 2.7 - Esbós

- **Verifica / Compila.** Comprova el vostre esbós buscant errors en compilar-lo; informarà de l'ús de memòria per al codi i les variables a l'àrea de la consola.
- **Pujar.** Compila i carrega el fitxer binari a la placa configurada a través del port configurat.
- **Càrrega mitjançant Programador.** Això sobreescriurà el carregador d'arrencada a la placa; haureu d'utilitzar *Eines* > *Grava* el carregador d'arrencada per restaurar-lo i poder tornar a carregar al port sèrie USB. Tot i això, us permet utilitzar tota la capacitat de la memòria Flash per al vostre esbós. Tingueu en compte que aquesta ordre NO cremarà els fusibles. Per fer-ho, s'ha d'executar una ordre *Eines* -> *Grava* el carregador d'arrencada.
- **Exporta binari compilat.** Desa un fitxer **.hex** que es pot conservar com a arxiu o enviar-lo

a la placa mitjançant altres eines.

- **Mostra la carpeta del Sketch.** Obre la carpeta d'esbossos actual.
- **Inclou la biblioteca.** Afegeix una biblioteca al vostre esbós inserint declaracions *#include* al començament del codi. A més, des d'aquest element de menú podeu accedir al gestor de biblioteques i importar biblioteques noves des de fitxers **.zip**.
- **Afegeix un fitxer ...** Afegeix un fitxer font a l'esbós (es copiarà des de la seva ubicació actual). El fitxer nou apareix en una nova pestanya a la finestra d'esbós. Els fitxers es poden esborrar de l'esbós mitjançant el menú de pestanyes accessible fent clic a la icona de triangle petit que hi ha a sota del monitor de sèrie al costat dret de la barra d'eines.

## 2.8 - Eines

- **Format automàtic.** Això formata molt bé el vostre codi: és a dir, sagnar-lo de manera que l'obertura i el tancament de les claus arrossades s'alineïn i que les declaracions dins de les claus queden més recorregudes.
- **Arxiva el programari.** Arxiva una còpia de l'esbós actual en format **.zip**. L'arxiu es col·loca al mateix directori que l'esbós.
- **Arregla la codificació i torna a carregar.** Corregeix possibles discrepàncies entre la codificació de mapes de caràcters de l'editor i altres mapes de sistemes de funcionament.
- **Monitor sèrie.** Obre la finestra del monitor sèrie i inicia l'intercanvi de dades amb qualsevol placa connectada al port seleccionat actualment. Normalment, es reinicia la placa, si la placa admet *Reinici per obertura del port sèrie*.
- **Plotter sèrie.** Aplicació que ens permet dibuixar gràfiques.
- **Placa:** Seleccioneu el tauler que feu servir.
- **Port.** Aquest menú conté tots els dispositius en sèrie (reals o virtuals) del vostre equip. S'actualitzarà automàticament cada vegada que obriu el menú d'eines.
- **Informació de la placa.** Dona informació de la placa connectada.
- **Programador.** Per seleccionar un programador de placa quan es programa una placa o un xip i no s'utilitza la connexió sèrie USB integrada. Normalment no necessitareu això.
- **Enregistra el Bootloader.** Els elements d'aquest menú us permeten gravar un carregador d'arrencada al microcontrolador d'una placa Arduino. Això no és necessari per a l'ús normal d'una placa Arduino o Genuino, però és útil si adquiriu un microcontrolador ATmega nou (que normalment no inclou un carregador d'arrencada). Assegureu-vos que heu seleccionat la placa correcta al menú Placa abans de gravar el carregador d'arrencada a la placa de destinació. Aquesta ordre també estableix els fusibles adequats.

## 2.9 - Ajuda

Aquí trobareu fàcil accés a diversos documents que inclouen el programari Arduino (IDE). Teniu accés a Introducció, Referència, una guia de l'IDE i altres documents localment, sense connexió a Internet. Els documents són una còpia local dels documents en línia i poden enllaçar al lloc web en línia.

**Cerca a la referència.** Aquesta és l'única funció interactiva del menú Ajuda: selecciona directament la pàgina pertinent a la còpia local de la referència per a la funció o l'ordre que hi ha sota el cursor.

## 3 - Estructura d'un programa de arduino i ordres bàsiques

### 3.1 - Objectius

- Comprendre l'estructura d'un programa arduino (sketch)
- Definir les estructures de blocs
- Presentar les primeres instruccions

### 3.2 - Estructura d'un programa Arduino.

Un programa o Sketch de Arduino consisteix en dues seccions o funcions bàsiques:

- *Setup:* Les seues instruccions s'executen només una vegada, quan s'arranca el programa en encendre Arduino o quan premem el botó de reset. Generalment inclou definicions i inicialitzacions d'ací el seu nom.
- *Loop:* Les seues instruccions es van executant en seqüència fins al final... I quan acaba, torna a començar des del principi fent un cicle sense fi.

Quan obrim el **IDE de Arduino** (o fem [Menú]\Arxiu\nou) ell ens escriu ja aquestes dues funcions. Note's que el principi de cada funció és indicat per l'obertura de clau “ { “ i la fi de la mateixa correspon al símbol de tancar claus “ } “. És imperatiu que a cada obertura d'una clau corresponga un tancament de clau. En successius capítols ampliarem aquest concepte.

De fet el conjunt d'instruccions contingudes entre una obertura i tancament de claus es diu **bloc** i és de cabdal importància a l'hora que el nostre **Arduino** interprete de l'una o l'altra manera les instruccions que li donem.

Ara com ara ressaltar les línies que apareixen dins dels blocs principals:

```
// put your setup code here, to run once  
// put your main code here, to run repeatedly
```

Qualsevol cosa que escriguem precedit per “ // “ son comentaris, i seran ignorats. És a dir podem deixar-nos missatges dins del codi, (que d'una altra manera donarien errors). El compilador ignorarà qualsevol cosa entre // i la fi de línia.

### 3.3 - Primeres instruccions en Arduino C++.

Sembla obligat en el món **Arduino**, que el primer programa que fem siga el blinking LED, i està bé perquè il·lustra algunes idees interessants quant a les seues possibilitats:

- La capacitat de Arduino per a interactuar amb el món extern. Una cosa bastant inusitada per als qui estiguen acostumats a la informàtica tradicional, on la potència de càlcul ha crescut de manera espectacular, però continua sent impossible (o quasi), influir en el món exterior.

- La senzillesa de l'entorn de treball. En contraposició a un sistema tradicional d'editor / compilador / linker.

Arduino pot relacionar-se de diferents maneres amb el món que li envolta, Començarem pels pins digitals que poden usar-se com:

- Entrades: Per a llegir informació digital del món exterior.
- Eixides: Per a activar un senyal al món exterior.

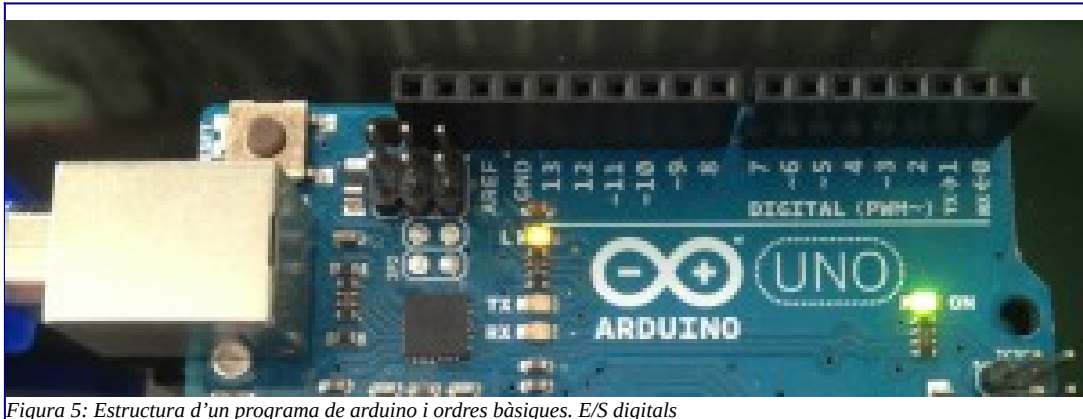


Figura 5: Estructura d'un programa de arduino i ordres bàsiques. E/S digitals

Arduino disposa de 14 pins que poden ser usats d'aquesta manera, numerats del 0 al 13.

En els exemples disposem d'un programa d'exemple que fa parpellejar un LED en la placa amb una cadència definida. Vegem com programar això (ja aprendrem a carregar els programes després).

Demanarem a Arduino que active el seu pin 13 com d'eixida digital i després encendrem i apagarem aquest senyal el que farà que el LED que té connectat de sèrie s'encenga o apague al ritme que marquem.

Per a indicar al sistema que desitgem usar el pin 13 com a eixida digital utilitzem la instrucció:

```
pinMode ( 13, OUTPUT ) ;
```

El primer paràmetre indica el **pin** a usar i “**OUTPUT**” és per a usar-lo com a eixida, i també podria usar-se el valor “**INPUT**” per a indicar que llegirem d'aquest pin.

Aquestes definicions es faran només una vegada al principi, en la funció setup(). La nostra quedarà, amb una única instrucció que declara que usarem el pin 13 com a eixida digital:

```
void setup()
{
    // // inicialitza el pin 13 com eixida digital
    pinMode( 13, OUTPUT) ;
}
```

- És important fixar-se en què malgrat ser una única instrucció, hem delimitat el bloc d'aquesta funció mitjançant obrir i tancar claus.
- Observe's que la instrucció finalitza en “;”. C++ obliga a acabar les instruccions amb un punt i coma que delimita l'ordre. Si s'omet generarà un error.

Per a encendre el LED usarem la instrucció:

```
digitalWrite( 13 , HIGH) ;
```

I una altra instrucció similar que li ordena apagar-ho:

```
digitalWrite( 13 , LOW) ;
```

El 13 indica el pin a utilitzar i HIGH, LOW indiquen el valor que desitgem posar en aqueixa eixida, que en Arduino corresponen a 5V per a HIGH i 0V per a LOW.

- Si en la funció loop() escriguérem aquestes dues instruccions seguides, Arduino canviaria aquests valors tan de pressa que no percebríem canvis, així que necessitem frenar-li una mica perquè puguem percebre el canvi.

Per a fer aquest retard de, diguem, un segon, utilitzarem:

```
delay(1000) ; // delay(n) "congela" Arduino n mil·lisegons
```

Per tant per a programar una llum que s'encén i s'apaga, hauríem de generar una seqüència d'ordres (*Com en una recepta e cuina*) que feren:

1. Informar a Arduino que utilitzarem el pin13 per a escriure valors( en el setup).
  2. Encendre el LED : Posar valor alt ( 5V) en aquest pin.
  3. Esperar un segon.
  4. Apagar el LED: Posar valor baix (0V) en aquest pin.
  5. Tornar a esperar un segon.
- *Si ometérem aquest segon retard, apagaria la llum i tornaria a començar trobant-se l'ordre de tornar a encendre. No apreciaríem que s'havia apagat. (No espere que em cregueu. Comproveu-ho).*
  - *El processador de Arduino UNO és molt lent des del punt de vista electrònic, però és capaç de commutar la llum ( passar d'encesa a apagat i volta a encendre) unes 15.000 vegades per segon.*

El primer concepte que heu de fixar, és que els ordinadors processen les ordenes en seqüència, una instrucció després d'una altra i en l'ordre en què li les doneu. *El nostre programa instrueix a l'ordinador perquè execute aqueixes instruccions i fixa l'ordre en el qual s'executen.*

La manera d'escriure un programa en Arduino C++ que faça l'anteriorment descrit és alguna cosa semblança a això :

[Codi: ARD\\_01.ino](#)

```
void setup()
{
  pinMode( 13 , OUTPUT); // Usarem el pin 13 com a eixida
}
void loop()
{
  digitalWrite(13 , HIGH); // Encén el LED
  delay(1000); // Esperar un segon
  digitalWrite(13 , LOW); // Apagar el LED
  delay(1000); // Esperar un altre segon
}
```

- Note's el sagnat de les línies per a destacar els blocs de codi. Això es considera bona

- pràctica i us ho recomanem encara més, perquè facilita molt la comprensió del programa.
- Quan us equivoqueu ( i creieu-me, us equivocareu) el sagnat ajuda, i molt, a visualitzar el programa.
- Només hi ha dos tipus de programadors. Els que s'equivoquen i els que s'equivocaran

Només ens falta ja, comprovar si hi ha errors i per a això premem la icona en blanc:



Figura 6: Estructura d'un programa de arduino i ordres bàsiques. Barra de funcions - Verifica

Si tot va bé, ( si no hi ha errors en roig) podem compilar i bolcar amb la següent fletxa, En cas contrari ( i creieu-me que us passarà amb freqüència) caldrà revisar els possibles errors i corregir-los. Tornarem sobre això en el futur.

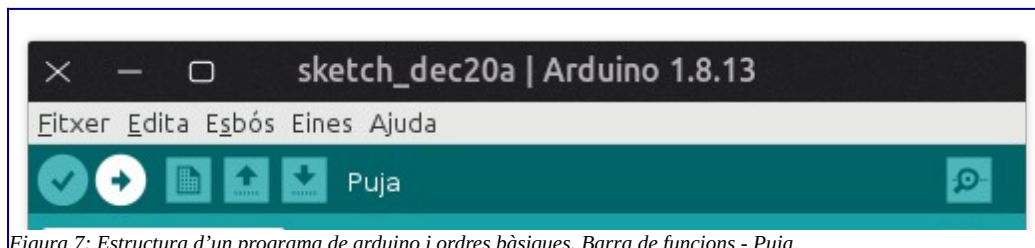


Figura 7: Estructura d'un programa de arduino i ordres bàsiques. Barra de funcions - Puja

La fletxa en blanc bolcara el nostre programa al Arduino i podrem comprovar que la llum del pin 13 parpelleja amb un retard d'un segon entre encesa i apagat.

- Suggeriment: Si modifiquem els valors del delay, modificarem la cadència del parpelleig.
- Nota: Això no funcionarà amb cap altre Pin del Arduino UNO, perquè només el 13 té un LED connectat.

### 3.4 - Resum

En aquesta lliçó hem après diverses coses importants:




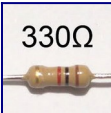

- El concepte clau d'un programa, com a seqüència d'instruccions que s'executa en l'ordre marcat.
- Hi ha dues funcions bàsiques en tot programa Arduino: setup() i loop()..
- Per a delimitar un bloc d'instruccions usem obertura i tancament de claus..
- Totes les instruccions acaben en punt i coma (Encara que hi ha excepcions)...
- Podem usar comentaris usant // .
- Hem après algunes instruccions inicials del Arduino C++.

## 4 - El nostre primer circuit

### 4.1 - Objectius

- Fixar algunes idees bàsiques sobre electrònica.
- Muntar un circuit amb LED i resistència i comprendre l'esquema elèctric.
- Aprendre com utilitzar la protoboard.
- Instal·lar el programa «Blinking LED»

### 4.2 - Material requerit

	Ardiuno Uno o compatible.		Un díode LED.
	Protoboard.		Una resistència de 330 Ohms ( $\Omega$ ).
	Cables Dupont.		

#### 4.2.1 - Algunes idees bàsiques sobre electrònica

Quan deixem fluir aigua d'un lloc alt a un altre més baix, l'aigua corre lliurement mentre no li ho impedim, i sempre de dalt a baix. Diem que les diferents altures suposen una diferència de potencial entre tots dos punts que pot ser transformada en treball útil.

Quan existeix una diferència de tensió elèctrica (o diferència de potencial) entre dos punts amb connexió, l'electricitat flueix del positiu (o de més càrrega) cap al negatiu o menys, i també podem obtenir treball útil d'aquest principi.

Encara que la física darrere d'aquests dos exemples és diferent, conceptualment són bastant semblants i per això parlem de:

- Corrent d'aigua / Corrent elèctrica.
- Cabal d'aigua / Intensitat de corrent.
- *Resistència al flux* / *Resistència elèctrica*.
- Capacitat d'una reserva d'aigua / Capacitat d'un condensador.

La idea és que el corrent elèctric flueix del positiu al negatiu perquè hi ha una diferència de tensió (que mesurem en Volts de símbol V) però això no és una mesura absoluta sinó la diferència que hi ha entre els punts en què ho mesurem.

- *De la mateixa manera, la diferència d'altura entre dos punts només representa això,*

una **diferència** i no indica a quina altura es troben respecte a una referència més o menys arbitrària.

Hi ha components que s'oposen a la lliure circulació del corrent. Els diem **resistències**, el seu valor es mesura en Ohms i el seu símbol és  $\Omega$ .

La **lei d'Ohm**, lliga tots aquests valors d'una forma precisa:  $V = R \times I$

On **V** és la tensió en volts, **R** la resistència i **I** la intensitat elèctrica que flueix.

- En el món de Arduino la tensió és quasi sempre 5V, que és la tensió al fet que funciona i la que és capaç de posar en les seues eixides digitals.

Una altra manera d'escriure aquesta lei d'Ohm és  $I = V / R$

El que implica que si la resistència del circuit és nul·la (o quasi, com en el cas d'un cable de coure) la intensitat del corrent es dispara i pot arribar a fondre el cable o component que trobe.

- Això es coneix com a **curtcircuit** o **curt simplement** i ha de ser evitat decididament ja que sol acabar amb olor a cremat i algun esglai, en el millor cas.

## 4.3 - El nostre primer circuit electrònic

En la sessió anterior programem el LED connectat al pin 13 del nostre Arduino. Hui duplicarem aquest circuit en l'exterior muntant-lo des del principi amb components discrets. El seu esquema elèctric seria:

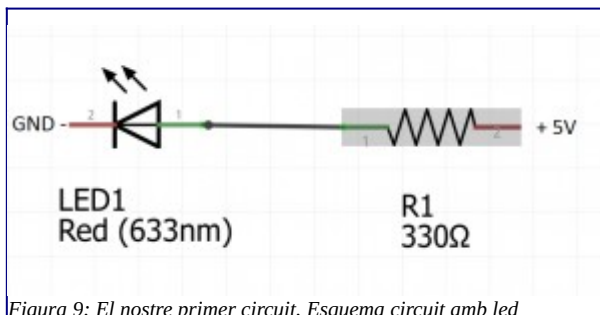


Figura 9: El nostre primer circuit. Esquema circuit amb led



Figura 8: El nostre primer circuit. Diode led

Veiem a l'esquerra el símbol del **díode LED** que és emissor de llum i per això té aqueixes fletxes sortints per a indicar-ho (LED ve del anglés Light Emitting Diode, o díode emissor de llum).

La resistència es representa per aqueix segon símbol indicant un nom R1 i el seu valor 330Ω.

Al seu torn veiem a l'esquerra les lletres GND per a indicar que és el negatiu. Té molts noms: Massa, El símbol -, Terra( encara que no és el mateix), Ground, Negatiu, càtode.

Finalment a la dreta el símbol de +5V indica l'extrem de tensió positiva o positiu i a vegades es representa com Vcc. Les línies rectes i negres indiquen connexió elèctrica mitjançant cables conductors.



- Un díode, és un component electrònic que només permet passar el corrent en una adreça. En la direcció del positiu al negatiu (la part ampla del triangle) al negatiu, la punta del triangle (que indica la direcció).
- Per a indicar quin de les potes d'un díode LED és el positiu, aquest sol ser de major longitud.
- Si es connecta al revés, tallarà el flux de corrent molt eficaçment i no s'il·luminarà en absolut.
- Les resistències en canvi no diferencien un extrems de l'altre, diem que no tenen polaritat.

És important entendre els esquemes electrònics perquè permeten comprendre amb rapidesa qualsevol circuit. Val la pena dedicar-li una mica d'esforç perquè són el llenguatge de l'electrònica. Una vegada comprés l'esquema elèctric del circuit, vegem la connexió en la Protoboard:

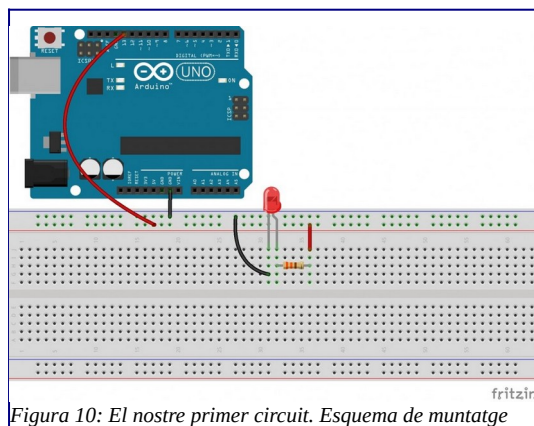


Figura 10: El nostre primer circuit. Esquema de muntatge

Aquest esquema segueix una pauta de marcar els cables que van a positiu en roig i els que van a GND en negre. Recomanem encara més se segueixca aquesta norma en la pràctica perquè ajuda a identificar possibles problemes i evita errors.

- La Protoboard uneix els punts de la línia blava entre si i els de damunt de la línia roja entre si, (se'n diu rails), però no connecta el raíl roig positiu amb el raíl negre negatiu.
- Al seu torn existeixen dues zones de línies verticals en la Protoboard. Aquestes línies verticals estan unides entre si internament, per a facilitar la connexió dels components, però no s'uneixen les línies paral·leles.

Les claus per a muntar el circuit amb èxit, són:

- Connectem el pin 13 de Arduino a la línia roja de la Protoboard: Positiu.
- Connectem el GND de Arduino a la línia blava de la Protoboard: Negatiu.
- Usem el raíl positiu (els pins de la línia roja) per a connectar a la resistència.
- L'altre extrem de la resistència es connecta al positiu del LED perquè estan en la mateixa vertical de la Protoboard (i aquesta els connecta elèctricament).
- Note's que el positiu del LED està clarament marcat com de major longitud mitjançant un xicotet angle prop de la base.
- Un díode LED quasi no presenta resistència pròpia, per la qual cosa sempre ha d'usar-se una resistència addicional que limite el pas de corrent, i evite que es creme. (Una resistència entre 220 i 330  $\Omega$  sol ser adequada).

- El circuit es tanca amb un cable des del negatiu del LED al raïl de negatiu.
- Quan el nostre programa pose un valor de HIGH (5V) en el pin 13 permetrà el flux de corrent pel circuit il·luminant el LED. Amb LOW senzillament el circuit estarà apagat, sense tensió.

Podem ara bolcar el programa que vam fer en la lliçó 3 (o  *simplement carregar l'exemple Blink*), seguint el procediment que definim allí, i veurem com aquesta vegada, a més del LED propi de Arduino, el nostre LED exterior parpelleja seguint el mateix cicle d'encesa i apagat.

## 4.4 - Resum



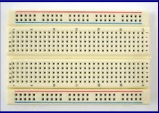



- Hem vist alguns conceptes bàsics d'electrònica: la llei d'Ohm, que relaciona la tensió la resistència.
- Hem identificat dos components bàsics en electrònica, resistències i els díodes.
- Vam aprendre a desxifrar els primers esquemes electrònics.
- Hem muntat el nostre primer circuit amb aquests components.

## 5 - Les entrades digitals

### 5.1 - Objectius

- Conèixer les entrades digitals.
- Llegir el primer polsador.
- Presentar els valors booleans.
- Un operador: Negació.

### 5.2 - Material requerit

	Ardiuno Uno o compatible.		Un díode LED.
	Una protoboard.		Dos resistència de 330 Ohms ( $\Omega$ ).
	Uns quants cables.		Un polsador

### 5.3 - Entrades digitals

Amb freqüència en electrònica necessitem saber si una llum està encesa o apagada, si algú ha premut un botó o si una porta ha quedat oberta o està tancada.

A aquesta mena de senyals tot / res, SI / NO, TRUE /FALSE, 0/1 se'n diu digitals, i podem manejar-les amb els pins de 0 al 13 de Arduino i per això parlem de pins digitals.

Molts dels sensors i actuadors que veiem en el món real són digitals:

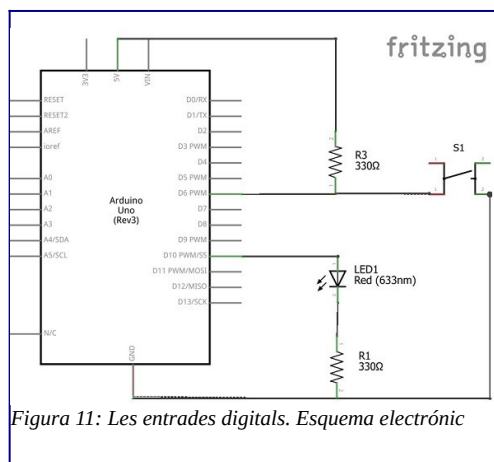
- *Com a actuadors digitals, tenim llums, alarmes, sirenes, desbloqueig de portes, etc.*
- *Com a sensors digitals podem esmentar botons i polsadors, Finals de carrera, desbordament de nivell, sensors de flames, fum o gasos tòxics.*

Hem vist que Arduino poden usar els **pins digitals** com a eixides tot o res per a encendre un LED. De la mateixa manera podem llegir valors, tot o res, del món exterior.

En aquesta lliçó veurem que els **pins digitals** de Arduino poden ser usats tant d'entrada com d'eixida. Llegirem un botó o polsador extern i encendrem o apagarem un LED en funció que el botó es prema o no.

## 5.4 - Esquema electrònic del circuit.

Muntarem un circuit amb un díode LED i resistència connectat al pin digital 10 de Arduino, tal com vam veure en les lliçons prèvies i a més un segon circuit amb un pulsador S1 connectat al pin 6 amb una resistència com es mostra en el diagrama següent.

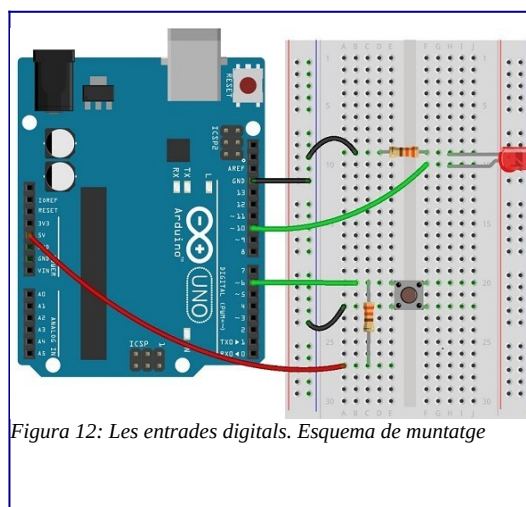


Observe's que mentre no premem S1 el pin 6 de Arduino està connectat a 5V a través de la resistència R3 forçant una lectura de tensió alta (HIGH). En canvi quan premem S1 tancarem el circuit del pin 6 a negatiu amb el que llegirà tensió baixa, LOW. En tots dos casos tenim un valor de tensió definit.

Si no posàrem la resistència R3, en prémer S1 llegiríem correctament LOW en el pin 6. Però en deixar de prémer S1 el pin 6 estaria en un estat flotant, que és ni HIGH ni LOW sinó indeterminat. Com això és inacceptable en circuits digitals forcem una lectura alta amb R3.

- A aquesta resistència que força el valor alt en buit se'l coneix com **pullup**. Si la connectàrem a massa per a forçar una lectura a negatiu se'n diria **pulldown**.
- Aquesta resistència és clau perquè les lectures del pulsador siguin consistents. El circuit, simplement, no funcionarà bé si s'omet (tornarem sobre això).

I ací tenim l'esquema per a protoboard del circuit.



- En aquest esquema hem seguit la pràctica habitual d'usar cables negres per a connectar a massa i cables rojos per a connectar a tensió (5V).

- *Observe's que el pulsador S1 té quatre pins (el que està sobre la resistència horitzontal). Això és perquè cada entrada de l'interruptor té dos pins connectats. En el nostre circuit simplement ignorem els pins secundaris.*

## 5.5 - Llegint els pulsadors

Comencem fent un programa que faci que el LED s'encenga quan premem el botó i s'apague quan el soltem. Per a això demanarem a Arduino que configure el pin digital 10 (D10) com a eixida per a manejar el LED, i el pin digital 6 (D6) com a entrada per a llegir el botó.

Normalment en programes senzills n'hi ha prou amb posar el número de pin en les instruccions. Però a mesura que el programa es complica això tendeix a provocar errors difícils de detectar.

Per això és costum definir **variables** amb els números de pin que usem, de manera que puguem modificar-los tocant en un sol lloc (i no havent de buscar al llarg del programa). Escriurem això una mica més elegantment:

```
int LED = 10 ;
int boto = 6;

void setup()
{
    pinMode( LED, OUTPUT) ;           // LED com a eixida
    pinMode( boto , INPUT) ;          // botó com a entrada
}
```

- *Atenció: C++ diferencia entre majúscules i minúscules i per tant LED, Led i led no són el mateix en absolut. De la mateixa manera, pinMode és correcte i en canvi pinmode generarà un error de compilador fulminant.*
- *He usat la variable boto sense accent perquè no és recomanable usar-los ni la ñ en els noms de variables, perquè poden passar coses estranyes.*

Vam veure que per a encendre el LED bastava usar **digitalWrite( LED, HIGH)**. Per a llegir un botó es pot fer una cosa similar: **digitalRead(boto)**. Vegem com podria ser nostre **loop**:

```
void loop()
{
    int valor = digitalRead(boto) ;           // llegim el valor de boto en valor
    digitalWrite( LED, valor) ;
}
```

Fàcil no? Encara que el LED està encès fins que premem el botó i s'apaga en prémer.

Com podríem fer el contrari, que el LED s'encenga en prémer i s'apague si no? Bastaria amb escriure en LED el contrari del que llegim en el botó.

Existeix un operador que fa això exactament l'operador **negació “ ! “**. Si un valor donat **x** és HIGH, llavors **!x** és LOW i viceversa.

- *Un operador és un símbol que relaciona diversos valors entre si, o que modifica el valor d'una variable d'una manera previsible.*
- *Exemples d'operadors en C++ són els matemàtics com +,-,\*, / ; i hi ha uns altres*

*com la negació ! o el canvi de signe d'una variable : -x. Anirem veient més.*

De fet aquest tipus d'operacions són tan freqüents que C++ incorpora un tipus anomenat **bool** o **booleà** que només accepta dos valors TRUE (cert) i FALSE (fals) i són completament equivalents a l'1 / 0, i al HIGH / LOW.

Aquest nou programa seria una cosa així:

```
void loop()
{
  bool valor = digitalRead(boto) ;      // llegim el valor de boto en valor
  digitalWrite( LED, !valor) ;          // Escrivim valor contrari en LED
}
```

Hem definit valor com bool, perquè podem usar el valor de tensió alt com TRUE i el valor baix com FALSE. SI el botó no està premut el D6 llegirà TRUE i per tant posarà LED a FALSE. En cas contrari encendrà el LED.

```
void loop()
{
  bool valor = digitalRead (LED) ;
  digitalWrite( LED, !valor) ;
  delay ( 1000) ;
}
```

De fet podríem escriure una variant curiosa del blinking LED usant l'operador negació, en només dues línies:

- *Podem llegir la situació actual d'un pin (ens retorna el seu estat actual), encara quan l'hàgem definit com a eixida, En canvi no podem escriure en un pin definit com a entrada.*

```
void loop()
{
  digitalWrite( LED , ! digitalRead( LED)) ;
  delay ( 1000) ;
}
```

- La primera línia llig la situació del LED i la inverteix, després escriu això en LED.
- *Les instruccions dins dels parèntesis s'executen abans que les que estan fora d'ells. Per això el digitalRead s'executa abans que el digitalWrite.*

## 5.6 - Resum

- Hem vist una manera de llegir senyals digitals en Arduino del món exterior a més de poder enviar-les:
  - x digitalRead( pin)
  - x digitalWrite( pin , valor)
- Hem conegut un nou component: el polsador.
- Coneixem un nou tipus en C++, el booleà i un nou operador de negació.

## 6 - Muntatge pull-up i pull-down

### 6.1 - Objectius

- Entendre la funció del muntatge pull-up i pull-down
- Diferenciar els efectes de cada muntatge

### 6.2 - Introducció

Pull-up i pull-down és un muntatge pel qual assegurem un valor alt o baix (*HIGH* o *LOW*) a la eixida del circuit. S'utilitza necessàriament amb els interruptors i contactes per no deixar mai una entrada sense connectar a cap lloc.

### 6.3 - Pull-up

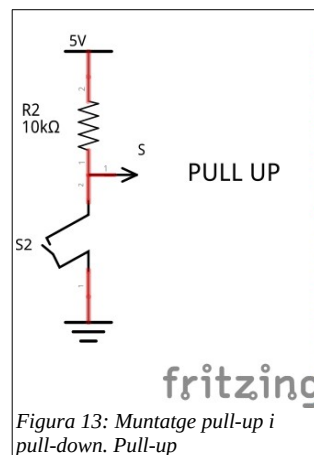


Figura 13: Muntatge pull-up i pull-down. Pull-up

Les resistències pull-up són resistències que s'utilitzen en circuits lògics per garantir un nivell lògic ben definit en un pin en totes les condicions. Com a recordatori, els circuits lògics digitals tenen tres estats lògics: alt, baix i flotant (o alta impedància). L'estat d'alta impedància es produeix quan el pin no s'arrossega a un nivell lògic alt o baix, sinó que es deixa "flotant". Una bona il·lustració d'això és un pin d'entrada no connectat d'un microcontrolador. No es troba ni en un estat lògic alt ni baix, i un microcontrolador podria interpretar de manera imprevisible el valor d'entrada com a màxim lògic o baix lògic. Si no hi haguera la resistència de tracció, l'entrada flotaria quan l'interruptor estiga obert i baixaria només quan l'interruptor estiga tancat.

## 6.4 - Pull-down

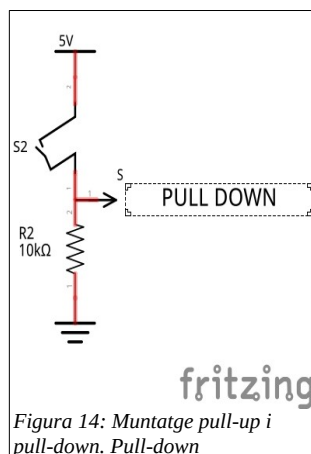


Figura 14: Muntatge pull-up i pull-down. Pull-down

Les resistències pull-down funcionen de la mateixa manera que les resistències pull-up, tret que porten el pin fins a un valor lògic baix. Es connecten entre terra i el pin corresponent d'un dispositiu. A la figura es pot veure un exemple de resistència desplegable en un circuit digital. Un interruptor de polsador està connectat entre la tensió d'alimentació i un pin del microcontrolador. En aquest circuit, quan l'interruptor està tancat, l'entrada del microcontrolador té un valor lògic elevat, però quan l'interruptor està obert, la resistència desplegable fa baixar la tensió d'entrada a terra (valor zero lògic), evitant estat indefinit a l'entrada. La resistència desplegable ha de tindre una resistència més gran que la impedància del circuit lògic, o bé podria ser capaç de baixar massa la tensió i la senyal d'entrada al pin es mantindria a un valor lògic constant, independentment de la posició del commutador.

## 6.5 - Resum

Les resistències pull-up no són un tipus especial de resistències; són simples resistències de valor fix connectades entre l'alimentació de tensió (generalment + 5V) i el pin adequat, cosa que resulta en definir la tensió d'entrada o sortida en absència d'un senyal de conducció.







## 7 - Les entrades analògiques

### 7.1 - Objectius

- Conèixer els potenciòmetres.
- Comprendre la conversió analògica a digital.
- Aprendre a usar les portes analògiques de Arduino.

### 7.2 - Material requerit

	Arduino UNO o equivalent
	Protoboard
	Cables connexió
	Un potenciòmetre

### 7.3 - Els potenciòmetres

Fins ara hem usat sempre resistències fixes, d'un valor donat. Però a vegades és convenient disposar d'un senyal variable per a controlar el circuit que ens interessa. Imagineu el volum d'un equip de música, o el dial que sintonitza una emissora en una ràdio FM.

Un potenciòmetre és, simplement, un mecanisme per a proporcionar una resistència variable.

Hi ha potenciòmetres de tantes grandàries, formes i colors, com pugueu imaginar, però al final són una resistència fixa d'un valor donat (10 k $\Omega$  en el nostre cas actual) i un mecanisme que permeti lliscar un dial conductor sobre aqueixa resistència, que ens permeti prendre una part d'aquell valor.

Per això un potenciòmetre sempre té 3 pins en fila. Els de l'extrem es comporten com una resistència del valor de fons d'escala del potenciòmetre, i un pin central que va prenent valors de resistència en funció del moviment que fem amb l'ajust.

Muntarem un circuit com aquest (en el qual el potenciòmetre aquesta retolat Pot1):

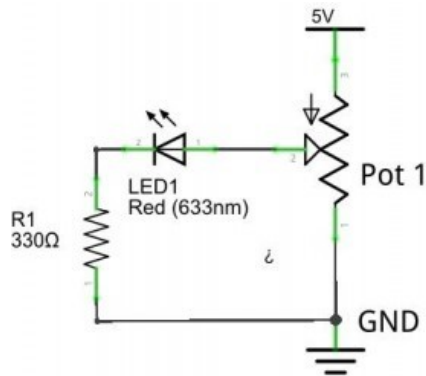


Figura 15: Les entrades analògiques. Esquema electrònic ús d'un potenciòmetre

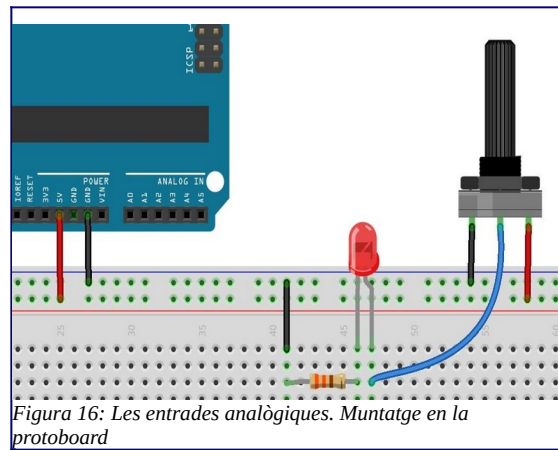
La idea és connectar 5V i GND als extrems del Potenciòmetre (no importa com és l'un i l'altre) i després connectar el pin central al positiu de un LED i el negatiu a GND directe, passant per una resistència de limitació.

D'aquesta manera quan girem el potenciòmetre estarem modificant la tensió que apliquem a l'entrada del LED, que variara entre 0 i 5V (Encara que ara sembla estrany és molt senzill) i haurem aconseguit un regulador d'intensitat del LED.

- Amb una resistència de 10k la intensitat en el circuit serà de:  $5V / 10.000\Omega = 0,5 \text{ dt.}$  Molt poc per a aconseguir il·luminar el LED que requereix uns 20 dt.. Així que durant la major part del gir del potenciòmetre el LED estarà apagat.
- Important: No oblidis la resistència R1. Encara que el potenciòmetre limite la intensitat, hi ha un moment en què arribara a zero i ací i el teu LED morirà en acte de servei.

## 7.4 - Circuit per a protoboard

El muntatge en la protoboard seria similar a això ja que utilitzarem el Arduino simplement per a donar tensió al circuit i res més, Veureu que la intensitat de la llum varia de manera contínua en girar el potenciòmetre.



- Recorda que a causa de l'excés de resistència del potenciòmetre de prova, durant la major part del gir de l'ajust el LED estarà apagat.
- Note's que en aquest cas utilitzem el nostre Arduino simplement com a font d'alimentació per a donar tensió al circuit.

## 7.5 - Arduino i les entrades analògiques

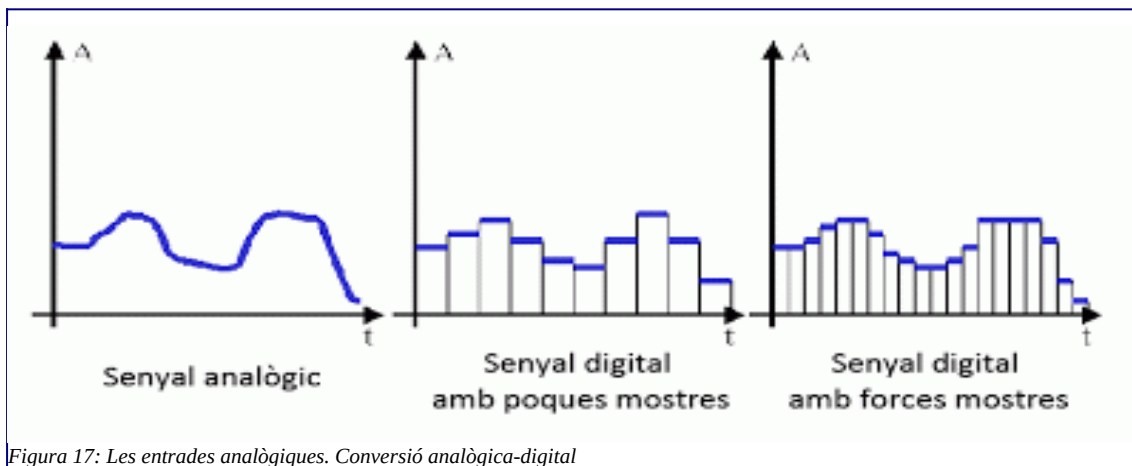
Amb Arduino hem vist que podem influir en el món exterior aplicant eixides tot / res en els pins digitals i també que usant PWM podem simular bastant satisfactòriament senyals analògics en alguns d'aqueixos pins.

També hem vist com detectar pulsacions de botons, definint com a entrades els pins digitals. Però en moltes ocasions els sensors que usem per a supervisar el món exterior, ens entreguen un senyal analògic. És el cas dels sensors de temperatura o distància, de pressió, d'intensitat de corrent en un circuit o de cabal d'aigua en una canonada.

Per a llegir aquest tipus de senyals continus necessitem un convertidor analògic a digital (o ADC per les seues sigles en anglés) i que ens permet llegir el valor d'un senyal analògic en un moment donat.

Aquests convertidors prenen una mostra del valor actual del senyal i ens entreguen el seu *valor instantani*, mesurat en Volts.

Mitjançant la lectura repetida de mostres al llarg del temps podem reconstruir el senyal original amb major o menor precisió, depenent de l'exactitud de la nostra mesura i de la velocitat a la qual puga prendre aqueixes mostres.

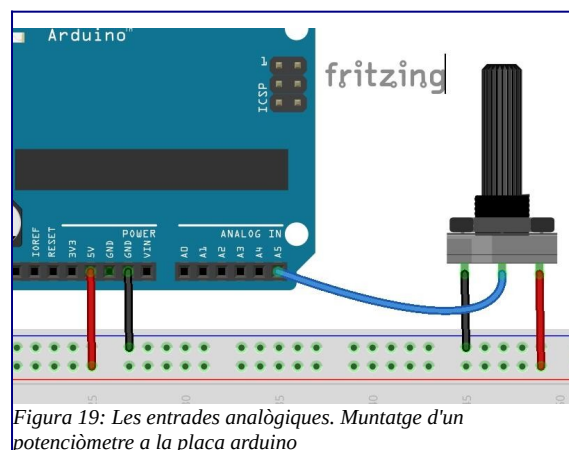


Arduino UNO disposa de sis convertidors analògic a digital, nominats d'A0 fins a A5, retolades com ANALOG IN:



Figura 18: Les entrades analògiques.  
Entrades analògiques

Vegem com usar les entrades analògiques amb un circuit com aquest, en el qual donem tensió als extrems d'un potenciòmetre i connectem el pin central (el variable) a l'entrada de la porta A5 de Arduino:



- Sembla bon moment per a destacar que els convertidors ADC llegeixen valors de tensió i no resistència, per tant, el que llegirem és **la caiguda de tensió** en el potenciòmetre a mesura que girem l'ajust.

La primera curiositat és que no necessitem declarar en el **setup()** que usarem una porta analògica. I la segona és que per a prendre una mostra (llegir) del pin A5, usarem la instrucció:

```
int Val = analogRead(A5) ;
```

- Els convertidors de Arduino UNO i Mega són de 10 bits de resolució pel que ens retornarà

valors entre 0 i  $2^{10} = 1.024$  per a tensions entre 0 i 5V. En canvi el Arduino DUE disposa de convertidors de 12 bits pel que el valor de les seues lectures estarà entre 0 i  $10^{12}$  o siga 4.096, és a dir té millor resolució (però només pot llegir fins a 3,3V).

- Assegura't de no usar sensors que puguin donar més de 5V màxim (amb Arduino UNO i Mega), ja que danyaries el xip principal de Arduino.

Escriurem un programa que llija el valor del pin A5 i l'envie a la consola perquè puguem visualitzar-lo.

## 7.6 - Usant les portes analògiques

Prova aquest programa:

```
//Codi: ARD_05_01.ino

void setup()
{
  Serial.begin(9600);    // Iniciem la porta serie
}
void loop()
{
  int Lectura = analogRead(A5) ;
  Serial.println( Lectura);
  delay(200) ;
}
```

Quan ho bolques, arranca la consola i voràs que a mesura que gires l'ajust les lectures varien de manera contínua reflectint la posició del potenciòmetre, les lectures reflecteixen la caiguda en volts.



Figura 20: Les entrades analògiques. Resultat al panel serie d'arduino

No puc resistir-me a proposar-vos aquesta prova: Desconnecta el potenciòmetre de la porta A5 i observa els resultats que Arduino envia a la consola. Perquè ixen aqueixos valors?

- Al no estar l'A5 connectat a cap referència vàlida, està surant i els valors que captura són mostra d'aqueixa incoherència. En realitat el que està fent la teua Duino és captar soroll aleatori de radiofreqüència i intentar donar-li sentit, però ho té malament, com podeu veure.
- No obstant això en condicions normals els valors que llegirà seràn relativament baixos. Vols que les oscil·lacions cresquen en valor?. Fàcil. Posa-li una antena. Val un simple cable de protoboard connectat des de l'A5 a res (O si agafes l'altre extrem entre els dits, tú mateix

*faràs d'antena).*

## 7.7 - Un últim comentari

Déiem anteriorment, que la fidelitat amb què podem mostrejar un senyal analògic depenia, bàsicament, de la resolució de la mostra i de la velocitat a la qual podíem mostrejar el senyal (Sample Rate en anglés).

Ja vam dir que la família Arduino, disposa de convertidors de 10 bits pel que la nostra resolució és de  $2^{10} = 1.024$  i en el cas del DUE de  $2^{12} = 4.096$ . Però fins ara no hem vist a quina velocitat podem prendre mostres amb el nostre Arduino. Ho comprovarem, amb aquest mateix circuit.

Tenim una funció anomenada **millis()** que ens indica en mil·lisegons el temps transcorregut des que iniciem Arduino i la podem usar per a veure quantes mostres podem prendre per segon.

```
//Codi: ARD_05_02.ino

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

void loop()
{
  unsigned long T ;
  int n = 0 ;
  T = millis();

  while (millis() <= T + 1000)      // Mentre no passe un Segon = 1000 ms
  {
    analogRead( A5) ;
    n++ ;                          // Comptem cada vegada que llegim
  }
  Serial.println(n);
}
```

- *Hem usat un unsigned long per a guardar millis perquè és el tipus que Arduino usa internament per al seu rellotge. Seria un error manejar millis amb un int perquè el seu valor màxim és 32.767 i mesurant mil·lisegons el comptador desbordaria en poca més de 32 segons.*

Si correu aquest programa en un Arduino UNO us donarà, si fa no fa, un resultat de 8.940 mostres o lectures per segon. No està malament.

És adequat per a mostrejar senyals que no varien massa ràpid amb el temps, com són quasi tots els sensors habituals en la indústria, però que es quedarà curt si voleu mostrejar senyals d'àudio.

- *Per a jugar amb àudio és millor usar un Arduino DUE. Té una velocitat de rellotge 4 vegades més ràpida (us farà falta), capacitat de mostreig a velocitat d'àudio (40 KHz) i autèntics convertidors DAC (digital to analog converters).*
- *De fet no és complicat augmentar la velocitat de mostreig fins a unes 20.000 mostres per segon amb un Arduino UNO, però per a això hem de pontejar Arduino i saltar a programar el xip interior Atmega 328. No és moment per a això, però hi ha formes.*

## 7.8 - Resum



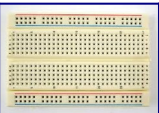


- Ja coneixem l'ús del potenciòmetre.
- Hem presentat els conceptes bàsics en la conversió analògica a digital.
- Vam aprendre a llegir les portes analògiques de Arduino.
- Sabem que podem llegir les portes analògiques unes 8.900 vegades per segon amb una resolució de 10 bits, és a dir entre 0 i 1.024.
- Vam conèixer la funció **millis()**.

## 8 - Les eixides analògiques

### 8.1 - Objectius

- Comprendre les diferències entre analògic i digital.
- Conèixer les eixides quasi analògiques de Arduino.
- Modulació PWM

### 8.2 - Material requerit

	Ardiuno Uno o compatible.		Un díode LED.
	Una protoboard.		Una resistència de 330 Ohms ( $\Omega$ ).
	Uns quants cables.		

### 8.3 - Analògic i digital

Totes les senyals que hem treballat fins ara amb el nostre Arduino, de entrada o de eixida, tenen una característica comú: són digitals, es a dir que poden prendre un valor ALT o BAIX, però no valor intermedis.

Si representem el valor d'un **senyal digital** al llarg del temps, veuríem alguna cosa així:

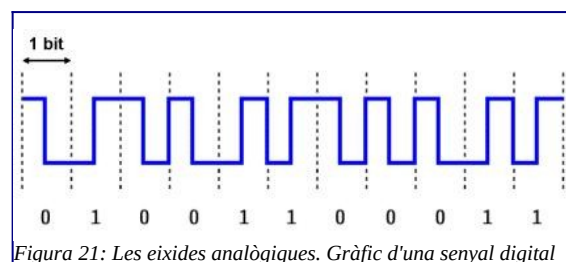
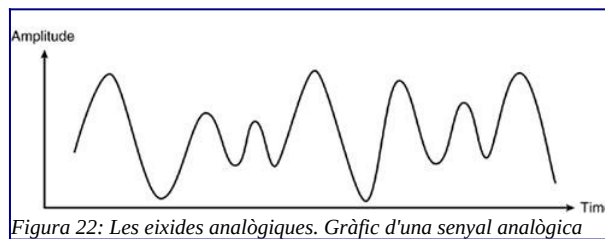


Figura 21: Les eixides analògiques. Gràfic d'una senyal digital

En la vida moltes coses son així, aprofes o suspens, encens la llum o l'apagues, però moltes altres son variables mesurables contínues i poden tenir qualsevol valor que imaginem, com l'angle de les agulles del rellotge o la temperatura, que dins de valors finits poden prendre tants valors intermedis com puguem imaginar.

A aquesta classe de variables les diem **analògiques** i una representació per contraposició al **digital**, seria una mica com això:





No és rar que vulguem controlar alguna cosa del món exterior amb un senyal analògic de manera que el comportament del sistema segueixca aqueix senyal. Podem per exemple voler variar la lluminositat d'un díode LED i no simplement apagar-lo o encendre'l.

A aquesta lliçó aprendrem a enviar senyals analògics als pins d'eixida de Arduino.

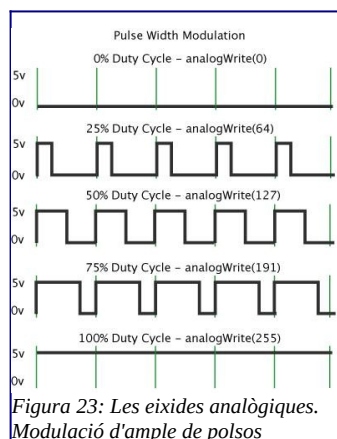
## 8.4 - Eixides quasi analògiques

Fins ara hem vist com activar les eixides digitals de Arduino, per a encendre i apagar un LED per exemple. Però no hem vist com modificar la intensitat de la lluentor d'aqueix LED. Per a això, hem de modificar la tensió d'eixida del nostre Arduino, o en altres paraules hem de poder presentar un valor analògic d'eixida.

Per a començar hem de deixar clar que els Arduino manquen d'eixides analògiques pures que puguem fer això (amb la notable excepció del Arduino DUE).

Però com els xics de Arduino són llestos, van decidir emprar un truc, perquè amb una eixida digital puguem aconseguir que quasi sembli una eixida analògica.

A aquest truc se'n diu **PWM**, sigles de Pulse Width Modulation, o **modulació d'ample de polsos**. La idea bàsica és posar eixides digitals que varien de forma molt ràpida de manera que el valor eficaç del senyal d'eixida siga equivalent a un senyal analògic de menor voltatge.



El sorprenent és que el truc funciona.

Fixar-vos en l'amplària del pols quadrat de dalt. Quant més ample és, més tensió mitjana hi ha present entre els pins, i això en el món exterior és equivalent a un valor analògic de tensió comprés entre 0 i 5V. Al 50% és equivalent a un senyal analògic del 50% de 5V, és a dir 2,5. Si mantenim els 5V un 75% del temps, serà l'equivalent a un senyal analògic de 75% de 5V = 3,75 V.

Per a poder usar un pin digital de Arduino com a eixida analògica, el declarem en el **Setup()** igual que si fóra digital:

```
pinMode( 9, OUTPUT) ;
```

La diferència ve a l'hora d'escriure en el pin:

```
digitalWrite(9, HIGH);    //Fica 5V en la eixida
digitalWrite(9, LOW);     //Fica 0V en la eixida
analogWrite( 9, V) ;      //analogWrite escriu en el pin de eixida un valor entre 0 i 5V,
                          //depenent de V (que deu estar entre 0 y 255).
```

D'aquesta manera si connectem un LED a una d'aquestes eixides PWM podem modificar la seua lluentor sense més que variar el valor que escrivim en el pin.

Però hi ha una restricció. No tots els pins digitals de Arduino accepten posar valors PWM en l'eixida. Solament aquells que tenen un símbol ~ davant del número. Fixar-vos en la numeració dels pins de la imatge:



Figura 24: Les eixides analògiques. Eixides analògiques arduino

- Solament els pins 3, 5, 6, 9, 10 i 11 poden fer PWM i simular un valor analògic en la seua eixida.
- Si intentes fer això amb un pin diferent, Arduino accepta l'ordre tranquil·lament, sense error, però per a valors de 0 a 127 entén que és BAIX i per a la resta posa ALT i segueix amb la seua vida satisfet amb el deure compliment.

## 8.5 - Modificant la lluentor d'un LED

Farem el típic muntatge d'una resistència i un díode LED, similar al de la lliçó 2, però assegurant-nos d'usar un dels pins digitals que poden donar senyals PWM. En la imatge he usat el pin 9.

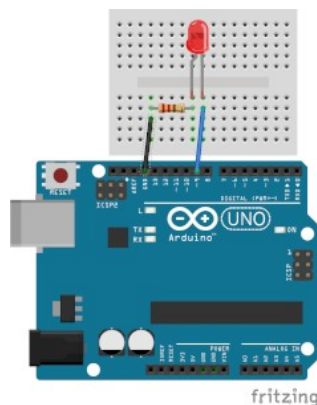


Figura 25: Les eixides analògiques. Esquema de muntatge eixida analògica

Podem escriure un programa semblant a això:

```
//Codi: ARD_06_01.ino

void setup()
{
  pinMode( 9, OUTPUT) ;
}
void loop()
{
  for ( int i= 0 ; i<255 ; i++)
  {
    analogWrite (9, i) ;
    delay( 10);
  }
}
```

El LED va augmentant la lluentor fins a un màxim i torna a començar bruscament. Podem modificar una mica el programa perquè la transició siga menys violenta:

```
//Codi: ARD_06_02.ino

void setup()
{
  pinMode( 9, OUTPUT) ;
}
void loop()
{
  for ( int i= -255 ; i<255 ; i++)
  {
    analogWrite (9, abs(i)) ;
    delay( 10);
  }
}
```

Hem fet el cicle de pujar i baixar la lluentor del LED amb un únic bucle. La funció **abs(num)**, retorna el valor absolut o sense signe d'un número **num**, i per això mentre que i viatja de -255 a 255, **abs(i)** va de 255 a 0 i volta a pujar a 255.

## 8.6 - Resum




- Descrivim a grans trets la diferencia ens valors digitals i valors analògics.
- Hem vist com simular valors analògics en una eixida digital de Arduino.
  - Només amb les eixides que ho accepten: pins 3, 5, 6, 9, 10 i 1.
  - Podem assignar valors entre 0 i 255.

## 9 - Comunicació amb l'exterior

### 9.1 - Objectius

- Comprendre la comunicació via port serie.
- Utilitzar la llibreria Serial.
- Operacions amb enters.
- Els tipus String i char.
- Operant amb Strings.
- La instrucció while.

### 9.2 - Material requerit

	Un portàtil o un PC
	Un cable serial per arduino
	Un arduino uno o equivalent

### 9.3 - Comunicació Sèrie amb el món exterior

Més abans que després, necessitarem comunicar el nostre Arduino amb el nostre PC. Les raons són varies, enviar-li ordres o rebre informació o senyals per exemple.

Els PCs disposen de teclats, pantalles i adaptadors de xarxa, però amb Arduino hem d'usar el port USB que establirà una **connexió en sèrie** amb el nostre PC.

La comunicació en sèrie és molt senzilla, basten dos fils per a enviar una diferència de tensió entre ells i poder marcar nivells alt (5V) i baix(0V) i amb això podem transmetre informació digital. Ara només ens falta pactar dues coses entre qui envia i qui rep:

- Un **codi comú** per a codificar els caràcters que enviem.
- Un **acord de velocitat** per a saber a quin ritme cal llegir les dades.

El codi comú que usarem amb Arduino es diu **codi ASCII** i és estàndard en tots els PCs. És una manera de codificar les lletres mitjançant números que representes aquests caràcters. Recordeu que només podem transmetre uns i zeros.

Així per exemple la lletra A se representa pel numere 65, la B el 66, C el 67... Pràcticament tots els PCs actuals utilitzen aquest codi i això inclou a Windows, Mac i Linux (i per això podem llegir emails enviats des de diferents plataformes), però és important comprendre que aquest és un més entre diversos codis de caràcters possibles (EBCDIC per exemple).

- *Actualment, en realitat, se sol usar una extensió del **codi ASCII** (anomenada **Unicode**) que permet a l'ús de caràcters no inclosos en la taula original, i que permet a representar caràcters com les Ñ, o accents per a l'espanyol, però també alfabetos diferents com el Kanji xinès o l'alfabet ciríl·lic. I aquest és el motiu pel qual podeu llegir les lletres xineses o russes en les pàgines d'internet d'aquests països.*

L'altre factor a pactar per a realitzar una comunicació sèrie és la velocitat. Atés que només disposem de dos fils per a transmetre, necessitem saber quan cal llegir la línia i això es fa establint un acord de velocitat. Si la velocitat d'enviament és diferent de la velocitat de lectura, el missatge final serà irrecognoscible.

Bona part dels errors de comunicació sèrie programant amb Arduino se solen deure a una diferència de velocitat entre l'emissor i el receptor.

Aquesta velocitat es mesura en bits per segon (bauds) i veurem que Arduino suporta diferents velocitats de comunicació sèrie.

## 9.4 - Establint la comunicació Sèrie

Arduino disposa d'una llibreria sèrie inclosa anomenada Serial, que ens permet enviar informació al PC i per a usar-la simplement hem de demanar-li en el nostre **setup()** que la incloga. La instrucció que s'encarrega és:

```
Serial.begin( velocitat ) ;
```

- *Note's que Serial té la S majúscules i que C++ diferencia entre majúscules i minúscules*

La velocitat és una valor entre 300 i 115.200 bits per segon. I sol ser costum establir-la en 9600 (el valor per defecte) però no hi ha cap raó per a això i aquesta no és una velocitat especialment alta.

Per a enviar un missatge des de Arduino al nostre PC podem usar les funcions **Serial.print()** i **Serial.println()**. Vegem un exemple:

```
int LED = 10 ; int boton = 6 ;
bool estat = false ;

void setup()
{
  Serial.begin(9600) ;      // Inicialitza el Port serial a 9600 bits per segon
}
```

```
void loop()
{
  int i = 54 ;
  Serial.println( i );
}
```

El **println()** enviara el valor d'i al port serie de Arduino (*repetidament*). Per a llegir-ho en el nostre PC necessitem un monitor de port serie. El IDE de Arduino inclou un molt senzill, però suficient que s'invoca amb el botó del monitor:



Figura 26: Comunicació amb l'exterior. Botó apertura monitor serie

Necessitem a més assegurar-nos que la velocitat de connexió és la mateixa en tots dos extrems. Fixa't en la part inferior dreta del monitor serie:

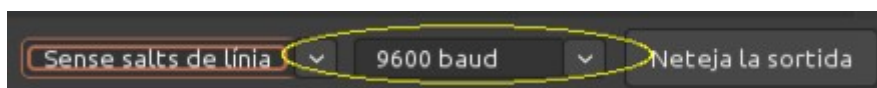


Figura 27: Comunicació amb l'exterior. Velocitat de comunicació port sèrie

Normalment la velocitat per defecte són els 9600 bits per segon o **bauds** en els quals hem programat la nostra porta serie, i si ho desplegau, veureu les diferents velocitats acceptables per a Arduino.

- *Estrictament parlant, bits per segon i bauds no són exactament el mateix excepte sota certes condicions particulars que en Arduino es compleixen, per la qual cosa ací podem usar-los com a sinònims.*
- *En el món Arduino sembla haver-hi un acord d'usar velocitats baixes com 9600 en lloc de més altes com 115200, per a evitar problemes. Això és alguna cosa que fa anys estava justificat per problemes de transmissió, però amb la tecnologia actual no hi ha motiu per a això. És més, quan necessitem utilitzar dispositius de comunicacions com a adaptadors Ethernet o BlueTooth per a comunicar-nos, la velocitat haurà de pujar necessàriament.*

Ara que sabem enviar informació i resultats al PC, veurem com podem operar amb enters i mostrar el resultat a la porta serie. En C++ els operadors numèrics són els normals en càlcul (i alguns menys freqüents):

- Addició: +
- Resta: -
- Multiplicació: \*
- Divisió sencera: /      Quocient sense decimals (ja que operem amb enters)
- Resta: %      Retorna la resta d'una divisió.

En C++ hem d'expressar les operacions matemàtiques en una sola línia i utilitzar parèntesi per a garantir que s'opera com necessitem. Anem amb alguns exemples:

Operació	Resultat	Comentari
int i = 4 * 2	resultat = 8	
int i = 4 * 2 / 3	resultat = 2	Perquè menysprea els decimals en ser sencer
int i = 14% 3	resultat = 2	La resta de 14 entre 3
int i = 2 + 8 / 2	resultat = 6	Calcula primer la divisió.
int i = (2+8) / 2	resultat = 5	El parèntesi força al fet que es realitzi primer la suma

Donada una expressió, la precedència d'operadors indica que operacions es realitzaren abans i quals després en funció del seu rang. Per als quals s'inicien en C++ no és fàcil saber que operadors tenen preferència, per la qual cosa és més segur que davant el dubte useu parèntesi.

Els parèntesis forcen les operacions d'una forma clara i convé utilitzar-los davant el dubte perquè d'una altra manera, detectar els errors d'operació pot tornar-se molt difícil especialment quan un comença a programar.

L'operadora resta és més útil del que sembla a primera vista perquè ens permet saber si un nombre és múltiple d'un altre. Suposem que volem saber si un número donat és parell.

Podríem escriure un programa com aquest:

```
void setup()
{
    Serial.begin(9600) ; // Inicialitza el Port serie
}
void loop()
{
    int i = 27 ;           //El número en qüestió
    if ( i % 2 == 0)
        Serial.println("És parell.") ;
    else
        Serial.println("És imparell");
}
```

Donant a i diferents valors podem comprovar com funciona l'operadora resta %. Tornarem sobre això quan vegem alguns exemples de com calcular nombres primers.

En aquest programa hem usat d'una manera diferent el **Serial.println()** passant-li una **String** de text entre cometes. **Serial.print()** envia el text ( entre cometes) que li posem però no fa salt de línia quan acaba. En canvi **Serial.println()** fa el mateix i inclou al final aqueix salt de línia.

```
void setup()
{
    Serial.begin(9600) ; // Inicialitza el Port serie
}
void loop()
{
    Serial.print("Bons ") ;
    Serial.print("Dies ") ;
    Serial.println("a tots.") ;
}
```

C++ disposa d'una mena de variables anomenades **Strings**, capaces de contindre textos. Podem operar amb elles simplement definint-les com qualsevol altra mena de C++:

```
void loop()
{
    int resultat = 25 ;
```

```
String s = " El resultat és: " ; // Note's que la S de String és majúsc.
Serial.print( s) ;
Serial.println( resultat);
}
```

Un tipus **String** es defineix simplement posant entre cometes dobles un text, i es pot operar amb elles d'una forma similar a com operem amb enters. Prova:

```
void loop()
{
    String a = "hola " ;
    String b = "a tots." ;
    Serial.println( a + b);
}
```

I també podem construir un **String** sobre la marxa així:

```
void loop()
{
    int resultat = 25 ;
    String s = "El resultat és: " ;
    Serial.println( s + String( resultat ));
}
```

On imprimim el resultat de concatenar s String, i la conversió d'un int a String (L'operador + afig un String al final d'un altre).

## 9.5 - Rebent missatges a través del port Sèrie

Fins ara només hem enviat missatges des de Arduino cap al PC, Però com rebem missatges en Arduino?

En primer lloc disposem d'una funció anomenada **Serial.parseInt()** que ens entrega el que s'escrui en el monitor serie convertit a enter:

```
void loop()
{
    if (Serial.available() > 0)
    {
        int x = Serial.parseInt();
        Serial.println ( x) ;
    }
}
```

Aquest programa simplement rep en x els números que ens teclegen en la consola (quan premem intro) i si és un text, l'interpreta com a zero.

Hem utilitzat una altra funció **Serial.available()** que és un booleà. Convé per costum comprovar que abans de llegir el port serie hi ha alguna cosa que ens han enviat. Si n'hi ha, available() és True i en cas contrari és False.

Per a llegir un **String** del port serie hem de complicar-nos una mica més i parlar del tipus **char**.

Un dels majors maldecap en iniciar-se en C++ és comprendre la diferència, antiintuïtiva, entre **char** i **String**. **Char** és un tipus que representa un únic caràcter i es defineix amb cometes simples, a diferència de String que necessita cometes dobles:

```
char c = 'a' ;
String s ="a" ;
```

Encara que sembla el mateix per a C++ són molt diferents.



Per a llegir una cadena des del port sèrie necessitem llegir un caràcter cada vegada i després muntar un String a partir d'ells, però abans, assegura't de seleccionar tots dos NL & CR en la part inferior del monitor serie, per a garantir que s'envia el caràcter de fi de línia:

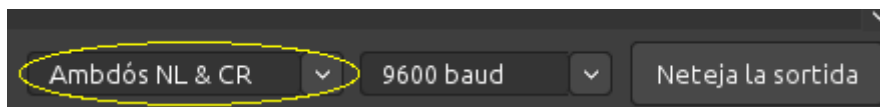


Figura 28: Comunicació amb l'exterior. Selecció NL & CR

Un programa per a llegir la consola seria una cosa així:

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

void loop ()
{
  char c = ' ';
  String missatge ="" ;
  if (Serial.available()) //Comprovem si hi ha alguna cosa esperant
  {
    while( c != '\n') //Si n'hi ha, ho llegim fins a l'intro
    {
      missatge = missatge + c ; // Afegim el llegit al missatge
      c = Serial.read(); //Llegir 1 caràcter
      delay(25);
    }
    Serial.println( missatge); //En eixir imprimir el missatge
    missatge = "" ; //Esborra-ho per a la pròxima vegada
  }
}
```

Ací usem una altra instrucció de C++ anomenada **while**. És similar a **if**, Executa repetidament el bloc que li segueix mentre es complisca la condició que li passem entre parèntesi:

```
while ( condició)
{ ..... }
```

Quan llig l'intro final del que escrivim, La condició `c != '\n'` es torna fals i ix del **while**.

D'altra banda, comprovem si hi ha una cosa disponible a la porta serie i en aquest cas muntem el missatge llegint un **char** cada vegada i sumant-li-ho a missatge per a construir un **String** que puguem imprimir en eixir.

- El motiu del `delay(25)` és que a una velocitat tan lenta, enviar un char de 8 bits per la porta serie, tarda molt més del que tarda Arduino a executar les instruccions del `while` i tornar a començar. Per això si se suprimeix el `delay` (i us recomane la prova) llegirà un caràcter bo (de la paraula escrita i com 10 caràcters fem per a un Arduino UNO o Mega).
- Si puguem la velocitat de comunicació a 115200 bits per segon, comprovàreu que no hi ha aquest problema ja que en multiplicar la velocitat d'enviament per més de 10 Arduino ja no té temps de tornar per més caràcters abans que arriben.

## 9.6 - Resum




- Hem vist com establir la comunicació amb el PC extern, tant per a enviar com per a rebre missatges sencers i de text.
- Hem presentat els tipus String i char.
- Hem vist les regles bàsiques per a operar amb enters i amb Strings.
- Presentem una nova instrucció: while.

## 10 - Crear gràfiques utilitzant el port serie

### 10.1 - Objectius

- Aprendre a usar el **Traçador Sèrie** per a fer **gràfiques** amb el IDE de Arduino.
- Incloure **diverses variables** en la gràfica.

### 10.2 - Material requerit

	Un portàtil o un PC
	Un cable serial per arduino
	Un arduino uno o equivalent

### 10.3 - Dibuixar una gràfica utilitzant el serial traçador

Ja hem utilitzat moltes vegades el monitor serie del IDE per a mostrar valors de les variables del nostre programa. Però en certes ocasions pot ser útil veure-ho en forma **de gràfica** en comptes d'en dades numèriques, per exemple, per a veure la tendència de les dades d'una forma senzilla i clara.

Perquè resulta que el IDE de Arduino incorpora des de la versió 1.6.6 una eina anomenada **Serial Traçador** que ens permet precisament crear gràfiques a partir de les variables que li indiquem. És molt senzilla d'usar i de moment no ofereix moltes opcions, però segurament vagen afegint característiques noves amb noves versions.

Per a utilitzar-la no hem d'aprendre res nou quant a la programació. Simplement farem un programa que mostre un valor pel port serie. Podeu utilitzar aquest programeta que genera números aleatoris cada cert temps:

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

void loop()
{
  int valor;
  valor = random(0,100);
```

```

Serial.println(valor);
delay(250);
}

```

Ara en comptes d'obrir el Monitor Sèrie, obrirem el Traçador Serial, que està en la barra d'eines, just davall.

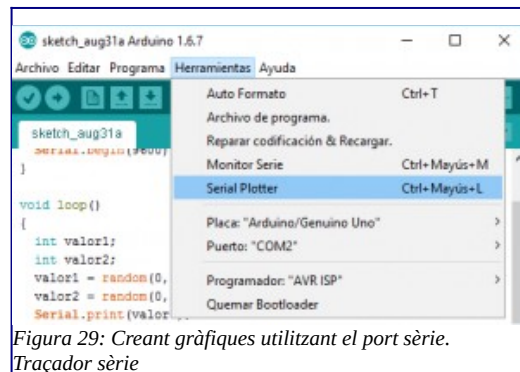


Figura 29: Creant gràfiques utilitzant el port sèrie. Traçador sèrie

I quan carreguem el programa en la placa, veurem com es va generant una **gràfica** a partir dels valors aleatoris que va agafant la variable.

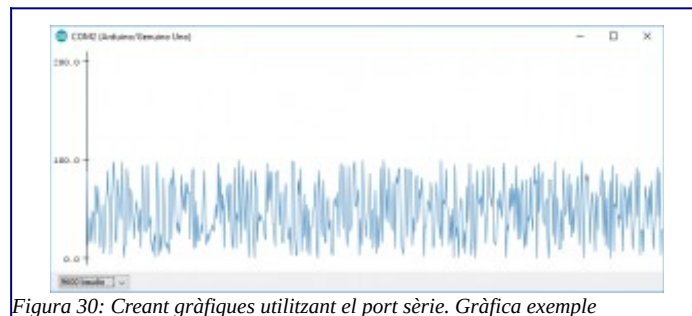


Figura 30: Creant gràfiques utilitzant el port sèrie. Gràfica exemple

## 10.4 - Com incloure més variables

L'eina també ens dona l'opció de mostrar diverses variables alhora en la mateixa gràfica. La manera de fer-ho és també molt senzilla, simplement haurem de treure les dues variables pel port serie però utilitzant un separador “,” entre ells, i automàticament els dibuixarà en la mateixa gràfica amb un color diferent.

Si afegim una altra variable que prengui també valors aleatoris al programa anterior, el programa quedaria de la següent forma

```

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

void loop()
{
  int valor1;
  int valor2;
  valor1 = random(0,100);
  valor2 = random(0,100);
  Serial.print(valor1);
  Serial.print(",");
  Serial.println(valor2);
  delay(250);
}

```

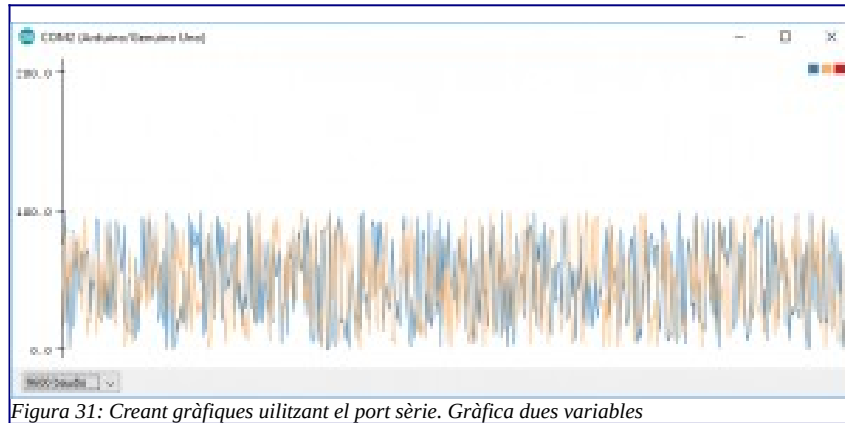


Figura 31: Creant gràfiques utilitzant el port sèrie. Gràfica dues variables

## Resum

- Hem descobert una nova eina que ens permet fer **gràfiques** fàcilment des del mateix IDE de Arduino.
- Sabem incorporar **diverses variables** a la gràfica.
- Ara que sabem que existeix aquesta eina i com s'utilitza, podeu incorporar-la als vostres projectes per a realitzar gràfiques de les mesures de sensors , la velocitat d'un motor, o el que se us passe pel cap.

## 11 - Annex 1. Alimentació de Arduino.

En aquest annex tractarem un tema fonamental per a aplicar a les plaques arduino en el món real: com podem alimentar el Arduino una vegada que aquest es desconnecta del port USB. Discutirem les opcions disponibles i brindarem consells útils sobre aquest tema endinsant-nos en els detalls del maquinari.

La forma més senzilla d'alimentar el arduino UNO és a través del port USB. No obstant això una vegada que desitgem col·locar la nostra placa arduino en la seua aplicació final, el port USB pot no ser la forma més òptima d'alimentar-la.

Desafortunadament la falta de coneixement o experiència en el tema d'alimentació pot portar a errors. Errors que al seu torn poden acabar en un funcionament inadequat o la destrucció del microcontrolador i/o la placa completa. L'objectiu d'aquest annex és brindar una guia que ens ajude a evitar aquests errors.

Volem aclarir en aquesta introducció que estarem tractant principalment amb les targetes que funcionen a 5 volts (mega, uno, duemilanove, leonardo, etc) i que ens referirem a aquestes en conjunt com “la placa arduino”, ja que totes comparteixen un disseny similar quant als seus circuits d'alimentació.

### 11.1 - Mecanismes d'alimentació per al arduino

Les plaques arduino més populars són molt versàtils i admeten diverses formes per a ser alimentades. En aquest apartat veurem detalladament cadascuna de les opcions que tenim per a donar-li poder a la nostra targeta. La següent imatge resumeix els mecanismes que podem utilitzar per a alimentar el Arduino UNO.

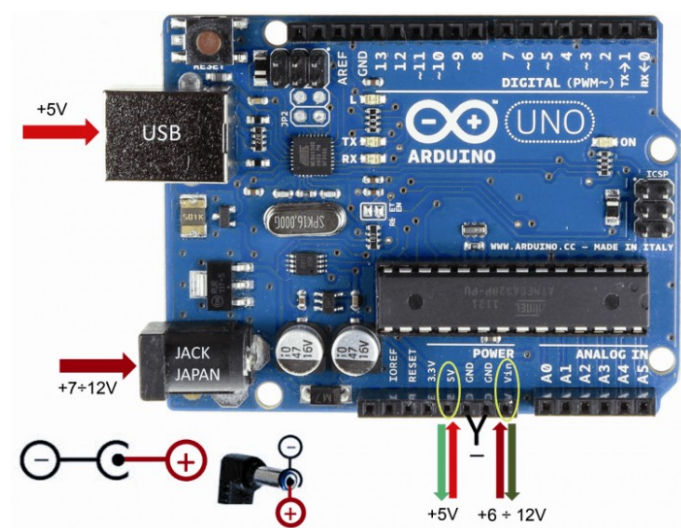


Figura 32: Alimentació de Arduino. Formes d'alimentar el arduino

## 11.2 - Alimentar el arduino mitjançant USB

Com ja esmentem, és la forma més senzilla d'alimentar el arduino. A través d'aquesta entrada s'admeten ÚNICAMENT 5 volts.



*Figura 33: Alimentació de Arduino. Placa i cable USB*

Podem obtenir els 5 volts del port USB de nostra PC i/o de qualsevol altre dispositiu compatible amb USB (com un televisor amb port USB, per exemple), d'un adaptador de telèfon mòbil amb eixida a USB o a través d'un dels “carregadors d'emergència” disponibles també per a telèfons mòbils.

El port USB compta amb un fusible PPTC que limita el corrent que el arduino (i els seus accessoris) poden demandar del port USB. El corrent màxim llavors queda limitada a uns 500 mA. Usualment no podem cometre errors de polaritat ni de voltatge quan usem USB per a alimentar.

### **Resum:**

Podem alimentar el arduino a través del connector USB quan s'utilitzen càrregues xicotetes i no es requereixen voltatges majors a 5 volts. En aquest cas no hem de preocupar-nos de la polaritat / voltatge, ja que és estàndard en tots els dispositius USB.

## 11.3 - El jack d'alimentació externa del arduino



*Figura 34: Alimentació de Arduino. Alimentador extern*

La placa arduino ve dissenyada per a acceptar alimentació mitjançant el jack estàndard que es troba en molts equips electrònics. Normalment s'utilitza un adaptador de corrent (AC/DC).

En tractar-se d'una entrada de corrent directe, la connexió de l'eliminador té una polaritat que ha de ser respectada: el pol positiu ha d'anar al centre del connector. El voltatge adequat a usar en aquesta entrada és de 7 a 12 volts DC.

Voltatges menors (5 a 7 volts) en aquesta entrada poden causar

que el regulador intern del arduino no pugui treballar correctament. Voltatges majors a 12 poden causar el ràpid sobrecalfament del regulador, encara que la quantitat d'accessoris connectats (la demanda de corrent) no siga gran.

Aquesta entrada té un díode de protecció per a inversió de polaritat, per la qual cosa si no es respecta la polaritat, no ocurreran danys, però la placa arduino NO funcionarà.

### 11.3.1 - Limitants del regulador de voltatge (sobrecalfament del arduino)

Un problema comú que podem trobar en alimentar el arduino a través del jack d'alimentació externa és que el regulador integrat en la placa pot sobrecalfar-se.

Per a entendre perquè succeeix això, hem de comprendre com funciona el regulador de voltatge NCP1117ST50T3G que es presenta en encapsulat SOT-223 en les plaques arduino. Aquest circuit integrat és un regulador de voltatge lineal, és a dir, un regulador que varia la seua resistència elèctrica interna per a mantindre un voltatge d'eixida constant en l'eixida.

En comportar-se aquest com una resistència elèctrica, tendeix a calfar-se de manera proporcional al corrent i al diferencial de voltatge entre la seua eixida i entrada. Per tant, si incrementem la diferència entre el voltatge d'entrada i el d'eixida, la potència dissipada en el regulador augmentarà.

Hem fet alguns càlculs bàsics i et vam mostrar a continuació el corrent màxim recomanat per a diferents voltatges d'entrada, assumint que permetrem una dissipació de 2 Watts en el regulador:

- Alimentació a 12 Volts:  $I = 2 / (12-5) = 2 / 7 = 285\text{mA}$
- Alimentació a 9 Volts:  $I = 2 / (9-5) = 2/4 = 500\text{mA}$
- Alimentació a 7 Volts:  $I = 2 / (7-5) = 2/2 = 1\text{A}$

Com podem veure, mentre més alt és el voltatge d'entrada, menor és el corrent que podem obtenir del regulador sense que aquest es calfe. Per a treballar en el punt més òptim, es requereix un adaptador AC/DC de 7 volts.

#### Resum:

El jack d'alimentació és la forma més segura d'alimentar el arduino a més de l'USB, no obstant això hem de tindre en compte la potència elèctrica que dissiparà el regulador de la targeta arduino. El rang de voltatge és de 7 a 12 volts i recomanem mantindre el regulador treballant amb un corrent apropiat segons el voltatge d'entrada (veure text a dalt).

## 11.4 - Alimentar el Arduino a través del pin Vin

El pin Vin, que es localitza en el grup de pins d'alimentació i terres compleix amb una doble funció:

- Permet aplicar una font d'alimentació externa en el rang de 12 a 6 volts DIRECTAMENT a l'entrada del regulador de la placa Arduino. En aquest cas, NO es compta amb protecció contra inversió de polaritat ni contra sobre corrent. **En cas d'aplicar voltatge directament al pin Vin, no s'ha d'aplicar simultàniament un voltatge en el jack.**



- Funciona com a eixida de voltatge quan el arduino s'està alimentant a través del jack d'alimentació. En aquest cas el voltatge present en Vin serà aquell que estiguem aplicant en el jack, restant la caiguda de tensió en el díode de protecció d'inversió de polaritat (al voltant de 0.7 volts). No es recomana connectar càrregues majors a 1000 dt. en aquest pin, ja que podem danyar el díode de protecció.

En tots dos casos el pol negatiu de l'alimentació estarà connectat a qualsevol dels pins etiquetats com GND.

#### **Resum:**

Podem utilitzar el pin Vin per a alimentar el arduino amb fonts no regulades de corrent directe o un conjunt de bateries AA o AAA (4 a 6 piles). L'alimentació amb bateries és recomanable a través d'aquest pin quan les nostres bateries proporcionen 6 volts, ja que no hi ha díode de protecció que cause caídas de tensió addicionals

## **11.5 - Alimentar el Arduino a través del pin 5V**

D'igual forma que Vin, el pin de 5 volts es pot usar de dues formes:

- Aquest pin funciona com una eixida de 5 volts per a altres circuits. El pin de 5 volts es connecta directament a l'eixida del regulador en la placa. Quan alimentem el arduino a través d'USB o el jack d'alimentació, l'eixida de 5 volts del regulador o USB és present en aquest pin.
- Podem utilitzar el pin de 5 volts per a alimentar directament el arduino amb una font de poder estabilitzada i regulada a 5 volts quan no hi ha un cable USB connectat o un adaptador de corrent connectat al jack.

#### **Resum:**

En alimentar el arduino a través del pin 5V no tenim protecció contra inversions de polaritat, sobre voltatge o curtcircuit, perquè tots els mecanismes de protecció en la placa es troben abans arribar a aquest punt del circuit. Només s'ha d'usar aquest mètode d'alimentació si planegem tindre una font d'eixida fixa (5V) i una connexió permanent amb el arduino (per a evitar errors).

## Taula de figures

Figura 1: ¿Què és i quines parts té arduino? - Pins de Arduino.....	4
Figura 2: L'entorn de desenvolupament integrat Arduino (IDE). Parts de l'IDE arduino.....	7
Figura 3: L'entorn de desenvolupament integrat Arduino (IDE). La barra d'eines.....	8
Figura 4: L'entorn de desenvolupament integrat Arduino (IDE). La barra de menús.....	8
Figura 5: Estructura d'un programa de arduino i ordres bàsiques. E/S digitals.....	12
Figura 6: Estructura d'un programa de arduino i ordres bàsiques. Barra de funcions - Verifica.....	14
Figura 7: Estructura d'un programa de arduino i ordres bàsiques. Barra de funcions - Puja.....	14
Figura 8: El nostre primer circuit. Díode led.....	16
Figura 9: El nostre primer circuit. Esquema circuit amb led.....	16
Figura 10: El nostre primer circuit. Esquema de muntatge.....	17
Figura 11: Les entrades digitals. Esquema electrònic.....	20
Figura 12: Les entrades digitals. Esquema de muntatge.....	20
Figura 13: Muntatge pull-up i pull-down. Pull-up.....	23
Figura 14: Muntatge pull-up i pull-down. Pull-down.....	24
Figura 15: Les entrades analògiques. Esquema electrònic ús d'un potenciòmetre.....	26
Figura 16: Les entrades analògiques. Muntatge en la protoboard.....	26
Figura 17: Les entrades analògiques. Conversió analògica-digital.....	27
Figura 18: Les entrades analògiques. Entrades analògiques.....	28
Figura 19: Les entrades analògiques. Muntatge d'un potenciòmetre a la placa arduino.....	28
Figura 20: Les entrades analògiques. Resultat al panel serie d'arduino.....	29
Figura 21: Les eixides analògiques. Gràfic d'una senyal digital.....	31
Figura 22: Les eixides analògiques. Gràfic d'una senyal analògica.....	32
Figura 23: Les eixides analògiques. Modulació d'ample de polsos.....	32
Figura 24: Les eixides analògiques. Eixides analògiques arduino.....	33
Figura 25: Les eixides analògiques. Esquema de muntatge eixida analògica.....	34
Figura 26: Comunicació amb l'exterior. Botó apertura monitor serie.....	38
Figura 27: Comunicació amb l'exterior. Velocitat de comunicació port sèrie.....	38
Figura 28: Comunicació amb l'exterior. Selecció NL & CR.....	41
Figura 29: Creant gràfiques utilitzant el port sèrie. Traçador sèrie.....	44
Figura 30: Creant gràfiques utilitzant el port sèrie. Gràfica exemple.....	44
Figura 31: Creant gràfiques uilitzant el port sèrie. Gràfica dues variables.....	45
Figura 32: Alimentació de Arduino. Formes d'alimentar el arduino.....	46
Figura 33: Alimentació de Arduino. Placa i cable USB.....	47
Figura 34: Alimentació de Arduino. Alimentador extern.....	47