Pràctica 0
Arduino i IDE

1. INTRODUCCIÓ

En aquesta pràctica ens familiaritzarem amb l'ús de la placa d'Arduino, que utilitza un microprocessador de tipus ARM ATmega328P (que utilitzarem en alguns apartats de les pràctiques de l'assignatura) i amb el seu entorn de programació que utilitzarem per programar-lo. Utilitzarem aquesta placa en properes pràctiques (excepte la pràctica 1) per programar un processador per anar controlant algun circuit i poder fer algunes aplicacions molt senzilles.
El primer que s'ha de fer abans d'arribar a aquest laboratori per fer aquesta pràctica és llegir el curt tutorial disponible al campus ('*Tutorial-Arduino_i_IDE.pdf*'). Dedicarem una estona al principi d'aquesta pràctica per resoldre dubtes del contingut d'aquest document. A continuació farem alguns exemples senzills disponibles a l'entorn de programació i els modificarem lleugerament com a exercicis.

IMPORTANT: Quan agafeu la placa, feu-lo pels costats de la placa, i no toqueu cap component amb els dits (amb la possible excepció del polsador, tot i que només serveix per fer un reset). L'electricitat estàtica pot fer-la malbé. Sereu responsables de la placa que preneu aquest dia, que serà la mateixa que tindreu en les properes pràctiques.
Quan connecteu el cable sèrie a la placa, feu-lo amb compte sense forçar la connexió.

2. OBJECTIUS

L'objectiu principal d'aquesta pràctica és conèixer molt bàsicament el funcionament de la placa d'adquisició i de l'entorn de programació per poder afrontar amb èxit els apartats corresponents a les properes pràctiques.

3. TREBALL PREVI

S'ha d'haver llegit el tutorial disponible al campus.

REALITZACIÓ PRÀCTICA.

A. Identificació dels elements de la placa

Descripció general: La placa principal de la placa consisteix en un microprocessador amb pins per poder interactuar amb el microprocessador externament i també amb alguns components connectats (led i interruptor)).

Components:

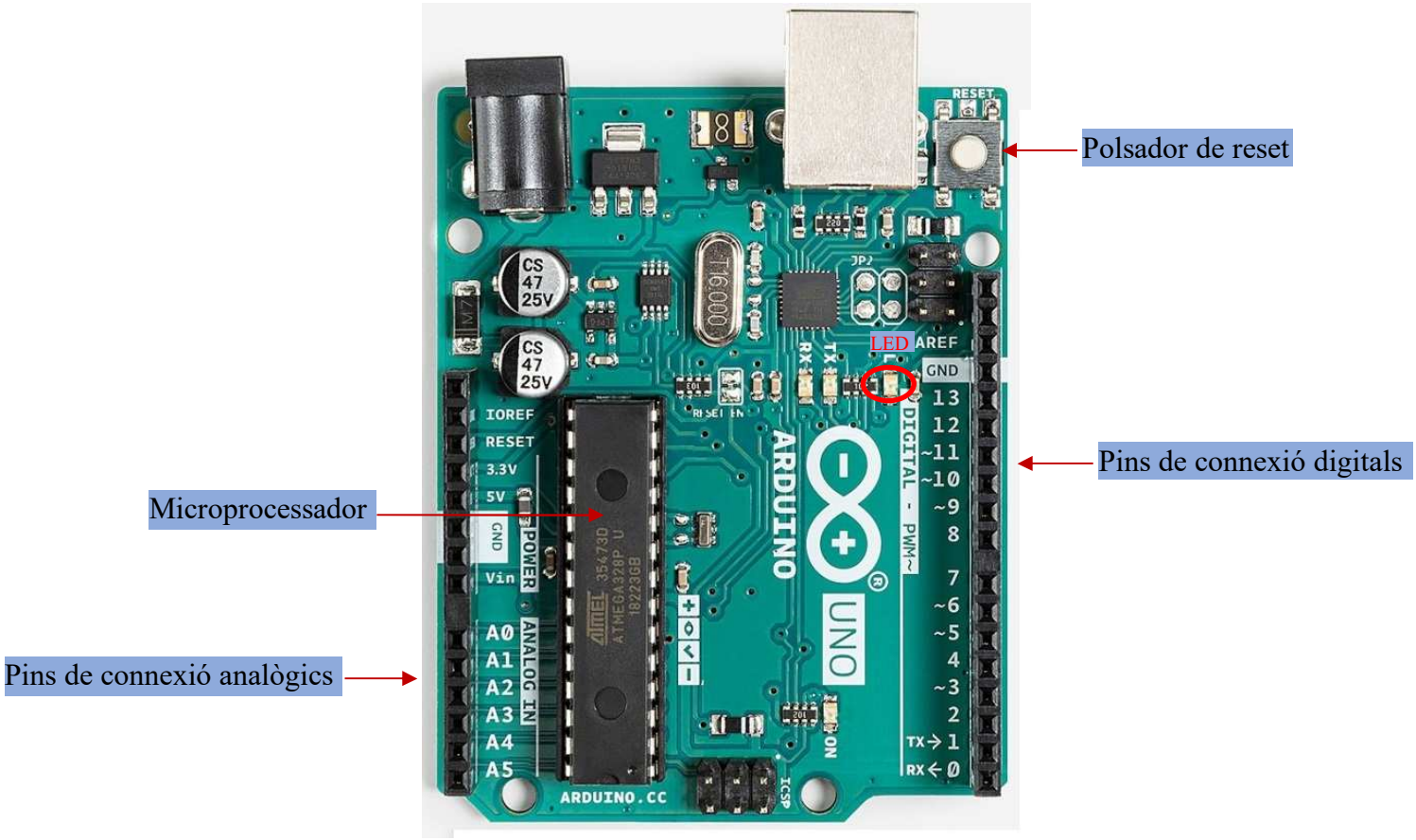


Figura 1. Components de la placa d'Arduino.

Relació de pins i les seves funcions (en aquesta assignatura només farem servir les funcions '*Dx*' i '*Ax*'):

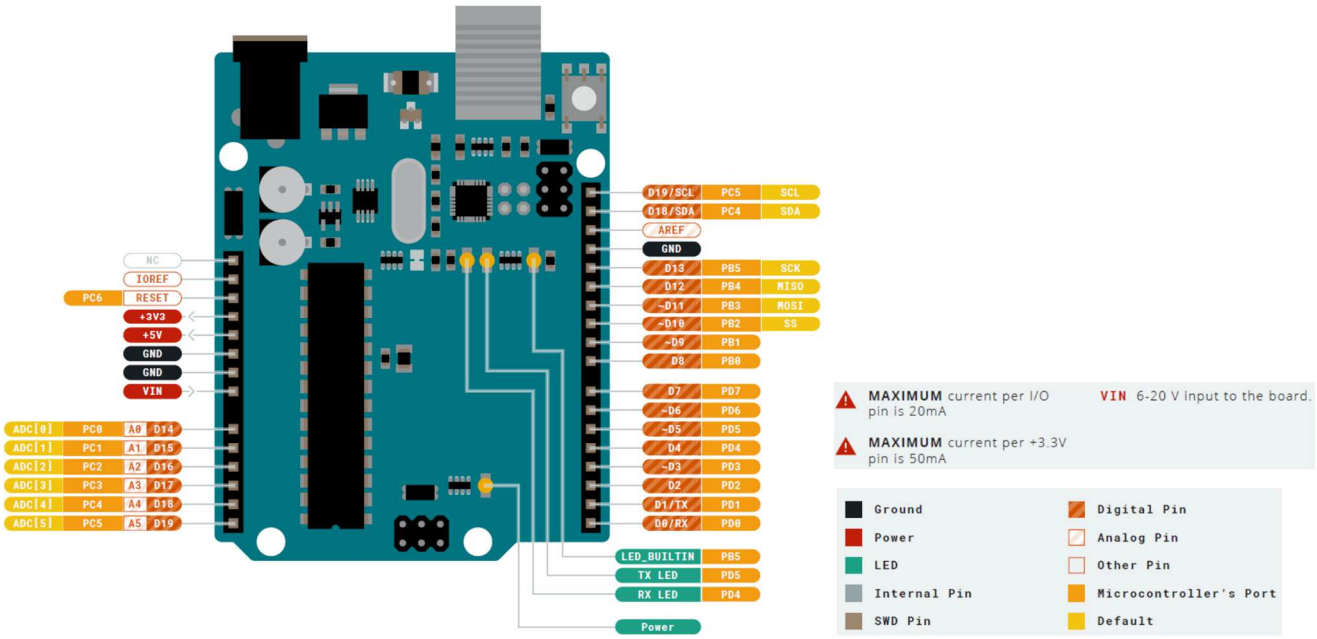


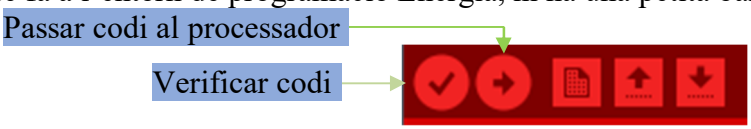
Figura 2. Relació de pins amb les seves funcions.

Hi ha un led connectat al pin 13.

Pel que fa als pins, nosaltres usarem només alguns dels següents: *Dx* (entrades o sortides digitals), *Ax* (entrades analógiques), *+5V* (tensió fixa de 5V), *+3.3V* (tensió fixa de 3.3V), *GND* (tensió de referència a la placa, que haurem de fer servir també com a terra dels nostres circuits per a què coincideixin).
S'ha de destacar també que alguns pins *Dx* tenen la indicació '*PWM out*', indicat amb el símbol '~'. Aquesta indicació vol dir que també podem generar en aquest pin un senyal periòdic quadrat de tensió (alternant 0V i 5V) a on podem controlar la part del període que està a 5V (el percentatge d'aquesta part respecte el període s'anomena '*duty cycle*' ($100 \cdot T_{on}/T$)).

B. Funcions habituals per les pràctiques

Pel que fa a l'entorn de programació Energia, hi ha una petita barra d'eines:



A les properes pràctiques, una de les coses que haureu de fer és desenvolupar una primera versió d'un codi abans d'arribar al laboratori (i, per tant, sense tenir la placa ni els circuits). En aquesta cas heu de fer servir 'Verificar codi' que comprova si ha problemes sintàctics al codi. Al laboratori, ja podrem passar el codi al processador i modificar el programa adientment per què acabi de funcionar correctament.
A l'opció 'Tools' del menu, hi ha l'opció 'Serial Plotter' que farem servir per visualitzar dades mesurades a una gràfica. Consulteu el tutorial del campus per veure com funciona.

Respecte al codi, el primer que s'ha de tenir clar és l'estructura general. Ha de tenir sempre dues funcions: '*void setup()*' i '*void loop()*'. La primera funció només s'executa una vegada (al principi quan executem el codi al processador; s'utilitza normalment només per configurar els recursos del processador), mentre que la segona s'executa després de haver executat setup() i, si arriba al final de d'aquesta funció, el codi es torna a repetir indefinidament des del principi. S'ha de tenir en compte això pel que fa també a la declaració de les variables. Consulteu també el tutorial del campus.

Control de les tensions als pins 'Dx' i 'Ax':

- *pinMode(pin,mode)*: Configura els pins, per exemple per especificar si és sortida o entrada (amb opció de pull-up) (*INPUT*, *OUTPUT*, *INPUT_PULLUP*). Exemple: *pinMode(19,OUTPUT)*.
  Retorna un valor entre 0 i 1024, que es correspon a 0V i 5V respectivament.
- *digitalWrite(pin,valor)*: Posa la tensió digital (HIGH o LOW) indicada a valor al pin. Exemple: *digitalWrite(19,HIGH)*.
- *digitalRead(pin)*: Llegeix la tensió digital al pin indicat. Exemple: *digitalRead(5)*.
  Retorna HIGH o LOW.
- *analogRead(pin)*: Mesura la tensió analògica que hi ha al pin indicat. Exemple: *analogRead(23)*.
  Retorna un valor entre 0 i 1024, que es correspon a 0V i 5V respectivament.
- *analogWrite(pin,value)*: Encara que la nostra placa no genera una sortida de tensió analògica, aquesta instrucció genera un senyal PWM al pin indicat (amb símbol '~') i amb un 'duty cycle' controlat per '*value*' (valor sencer entre 0 i 255, corresponent a duty cycle de 0% i 100% respectivament). Exemple: *analogWrite(40,150)*.

Control del temps:

- *delay(t)*: Atura l'execució del programa durant el temps *t* indicat (en mil·lisegons). Exemple: *delay(1000)*.
- *millis()*: Ens proporciona el temps en mil·lisegons transcorreguts des de que es va començar a executar el codi.

C. Us d'entrades i sortides digitals (Dx)

Pels exemples que veurem en aquest apartat no fa falta encara conèixer els circuits. Només hem de saber que posar un '1' lògic al pin (vol dir posar 3.3V en aquest pin respecte el terra) a on connectem un led farà que s'encengui, i quan posem un '0' lògic (vol dir posar 0V en aquest pin respecte el terra) farà que s'apagui. Pel que fa a un polsador, podem saber si s'ha polsat o no mirant la tensió que té el pin. Si aquesta tensió correspon a un '1' lògic és que no està polsat, i si és un '0' lògic és que està polsat.

1) Obriu l'exemple 'BlinkWithoutDelay' des de '*File → Examples → 02.Digital*'. Analitzeu el programa per entendre el que està fent a totes les instruccions. En lloc de LED_BUILTIN, utilitzeu el valor numèric del pin del led (13) directament. Fixeu-vos que podeu declarar variables globals fora de les funcions.
2) Correu l'exemple anterior i comproveu que el resultat és que un led s'encén i apaga cada cert temps constantment.
3) Al codi anterior, afegiu el codi d'aquí abaix després de la instrucció digitalWrite(). Analitzeu a priori (només veient el codi) quin serà l'efecte i correu el programa. Teniu en compte que haureu de declarar les variables globals '*control*' (inicialment a 0) i '*interval2*'; utilitzeu els valors interval=10 i interval2=1. Noteu que la intensitat lluminosa del led és molt menor que abans.

```
  control=0;
 }
 else {
  if ((currentMillis - previousMillis > interval2) && (control==0)) {
   digitalWrite(ledPin, LOW);
   control=1;
   }
 }
```

4) Obriu l'exemple 'Button' (el trobareu al mateix lloc que a l'apartat 1). Analitzeu de nou el programa per entendre com funciona.
5) Heu de connectar un polsador al pin 2 digital (el professor us indicarà com). Executeu l'exemple anterior i comproveu que el resultat és que un led s'apaga quan polsem l'interruptor corresponent.
6) Feu un programa (partint de l'anterior) que encengui els led només quan s'ha polsat durant més de 2 segons. Feu la comprovació dels polsadors cada 10ms aproximadament (feu servir la funció *delay()* i porteu un comptador per conèixer el temps total).
7) Opcional (si teniu temps): Feu un joc senzill: Un jugador polsarà el polsador durant un cert temps (però sempre menor de 5 segons). A continuació, el segon jugador hauria de polsar-ho durant el mateix temps (amb un cert marge d'error). Enceneu el led de la placa quan un dels jugador ha polsat, i que s'apagui quan l'ha deixat de polsar. Aneu mostrant a una gràfica (cada 100ms per punt) dues corbes:
   - Valors 100 ó 0 segons si el polsador corresponent està polsat o no respectivament.
   - Percentatge d'error que s'ha comès (poseu 0 mentre no es sàpiga el valor, i després manteniu-lo).