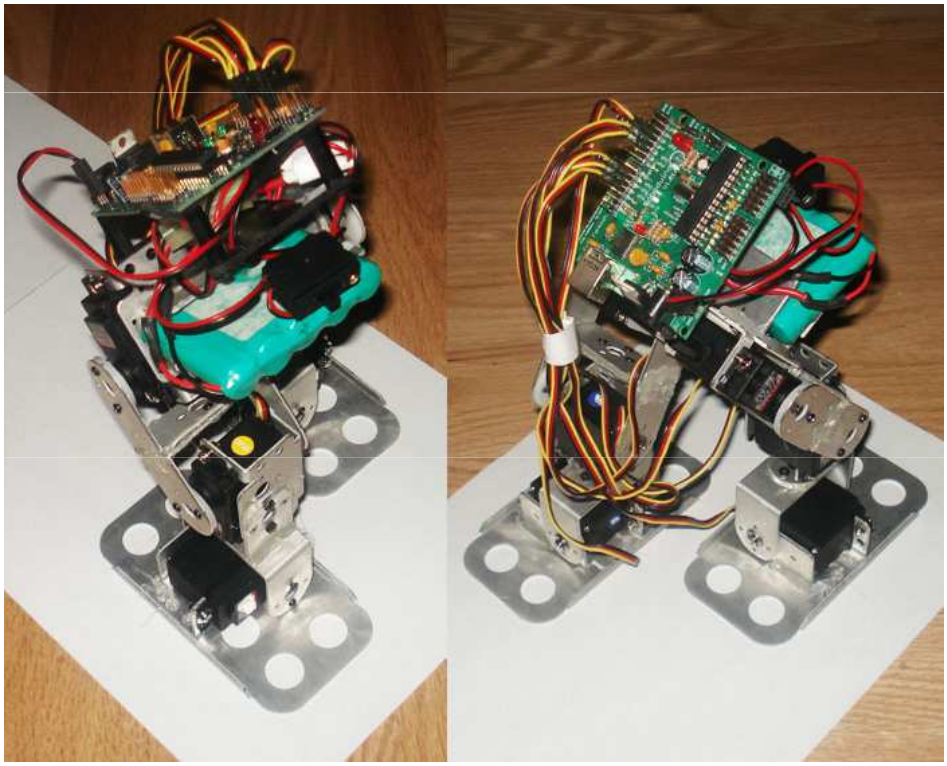
INICIACIÓN AO CONTROL PROGRAMABLE CON ARDUINO

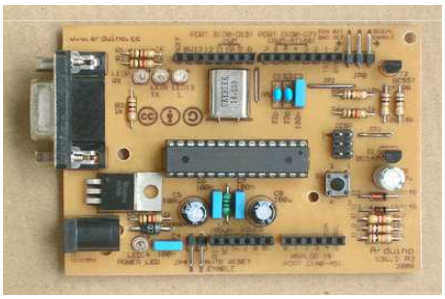


VANTAXES:

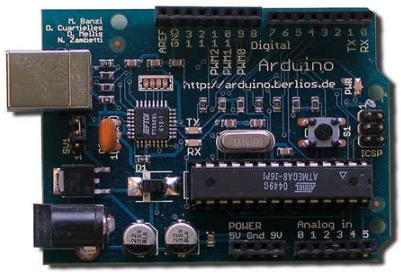
- **Hardware e software libre.**
- **Acceso ao seu código fonte (esquemas, planos, documentos, etc.)**
- **Programable en linguaxe de alto nivel.**
- **Campos de aplicación:**
  - **Educación.**
  - **Automatismos e domótica.**
  - **Música e arte.**
  - **Control programable e robótica.**
- **Información e descargas: http://arduino.cc**

EVOLUCIÓN

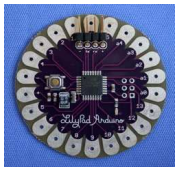
ARDUINO RS232



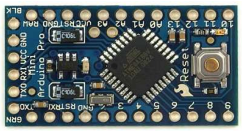
ARDUINO USB

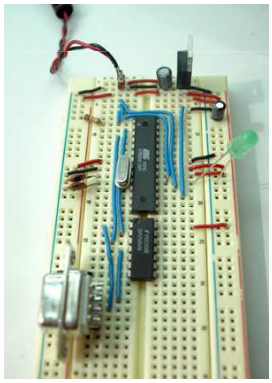


LilyPad



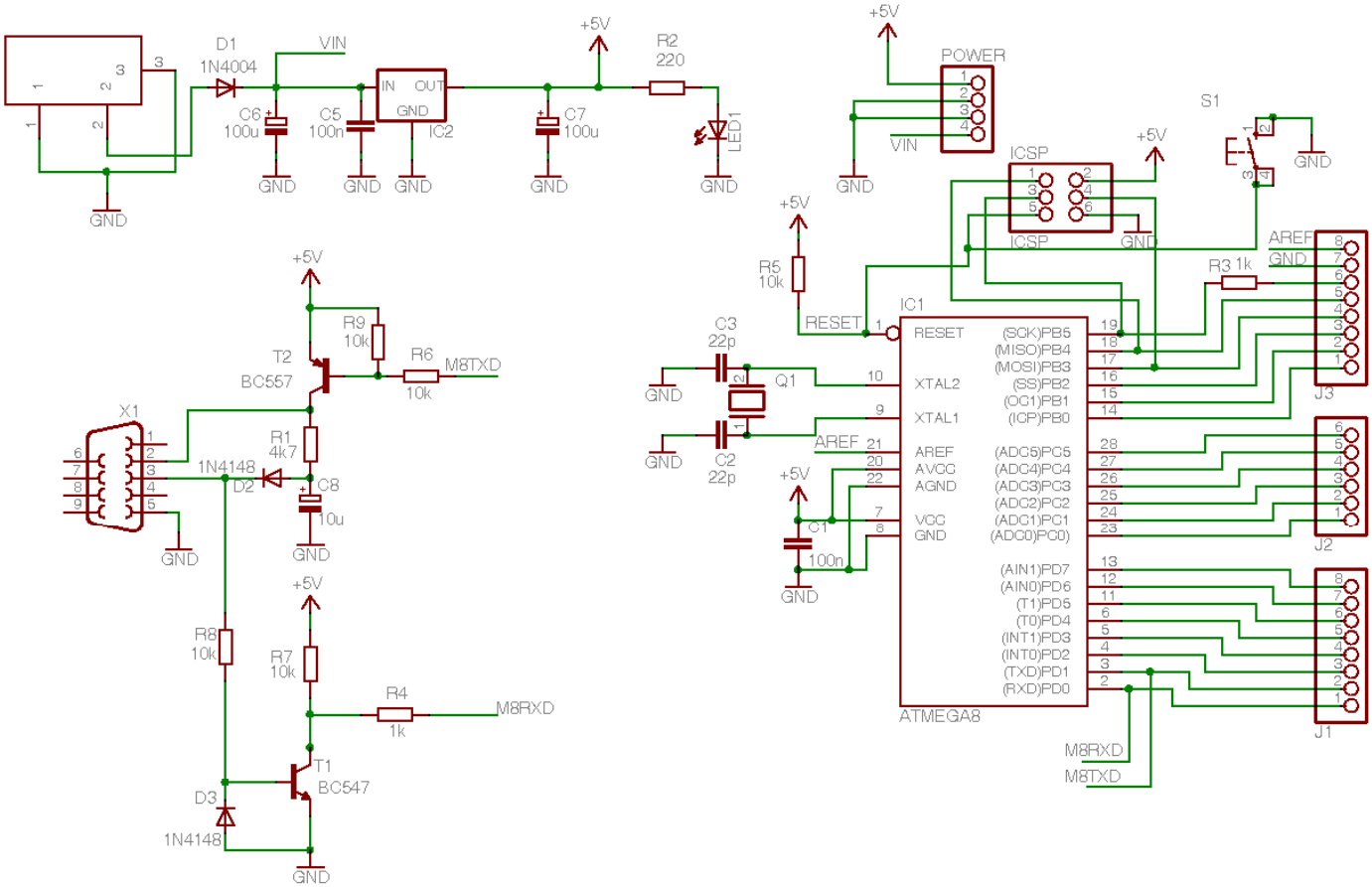
Pro-Mini



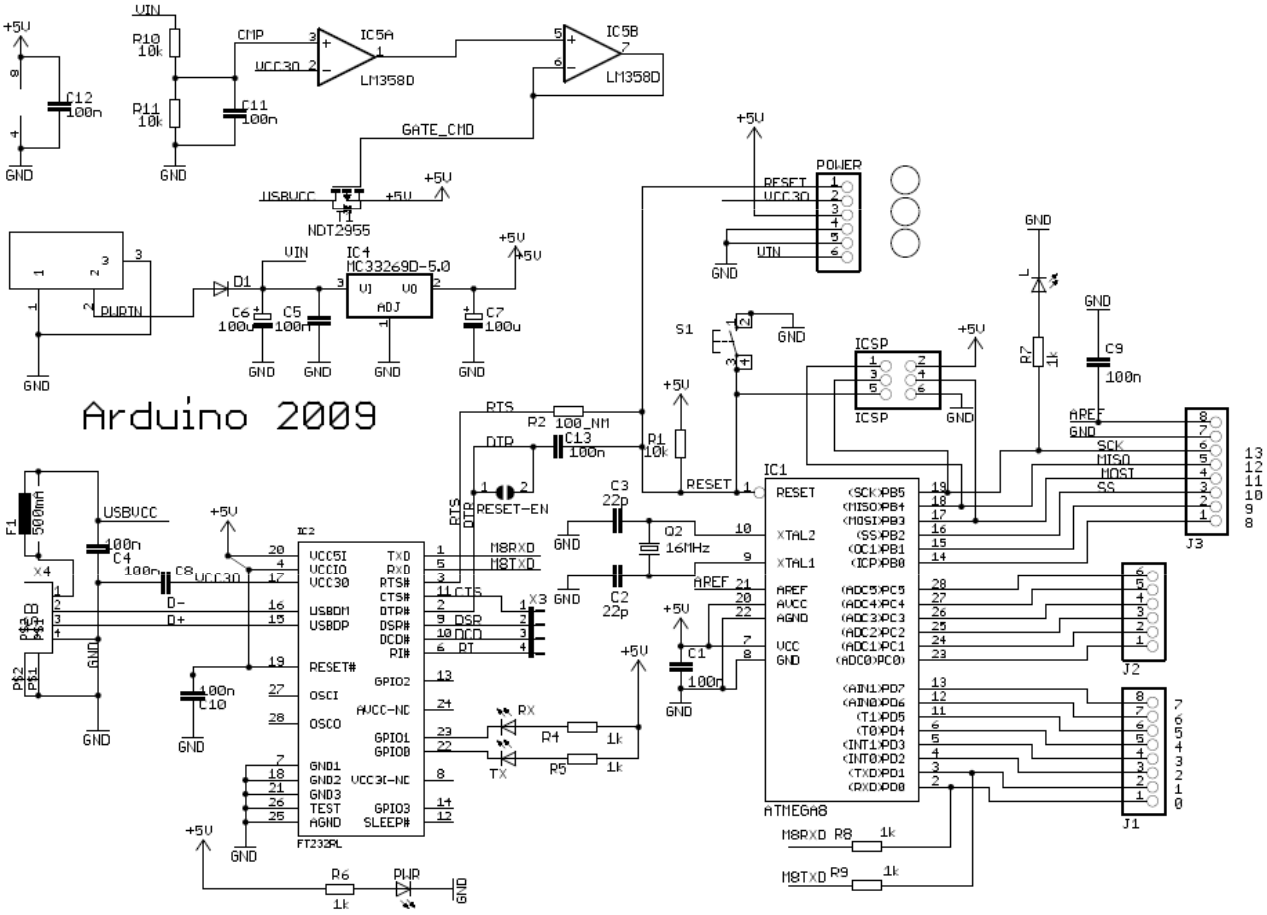


STAND-ALONE

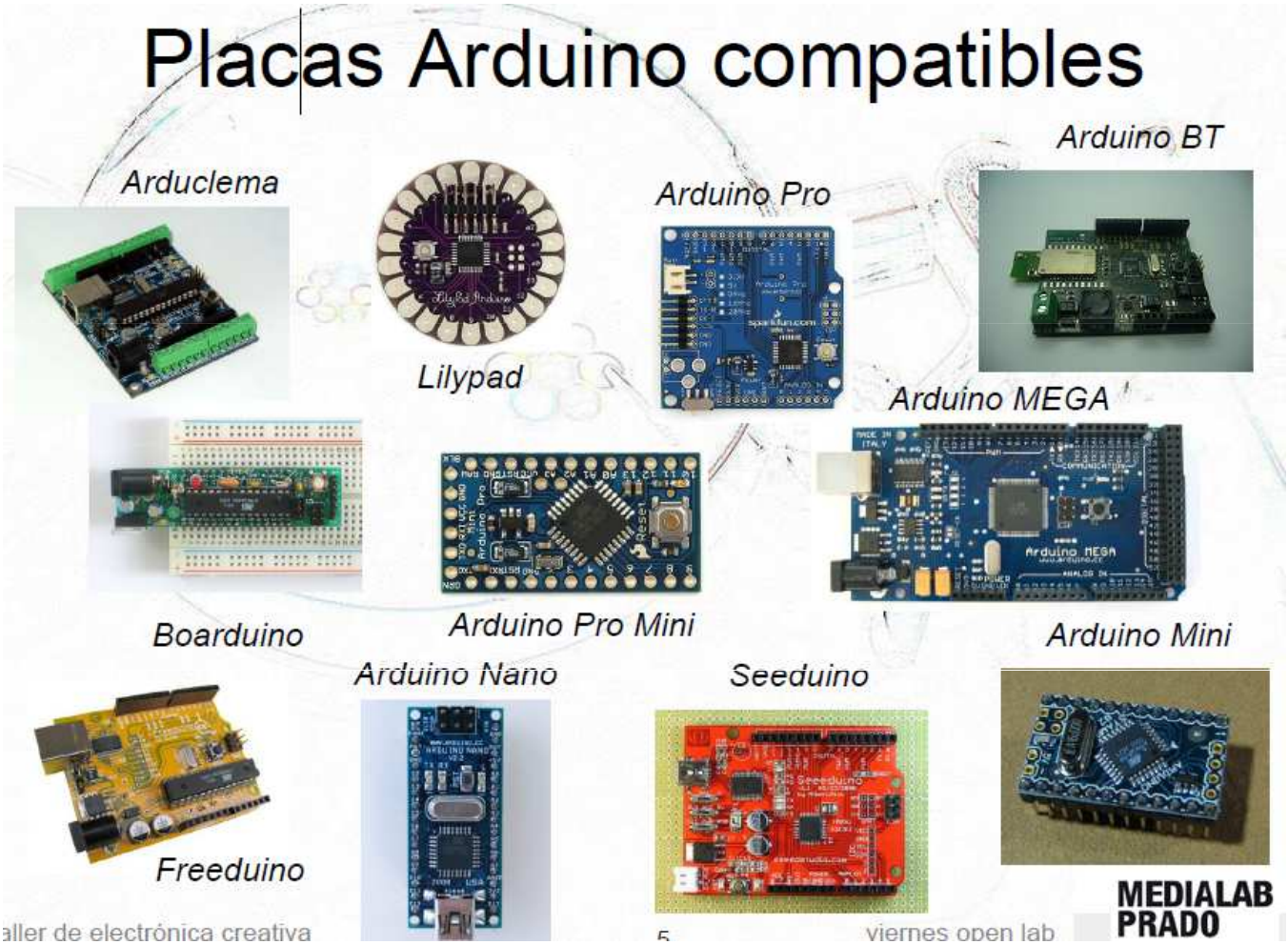
Esquema Arduino RS-232



Esquema Arduino USB

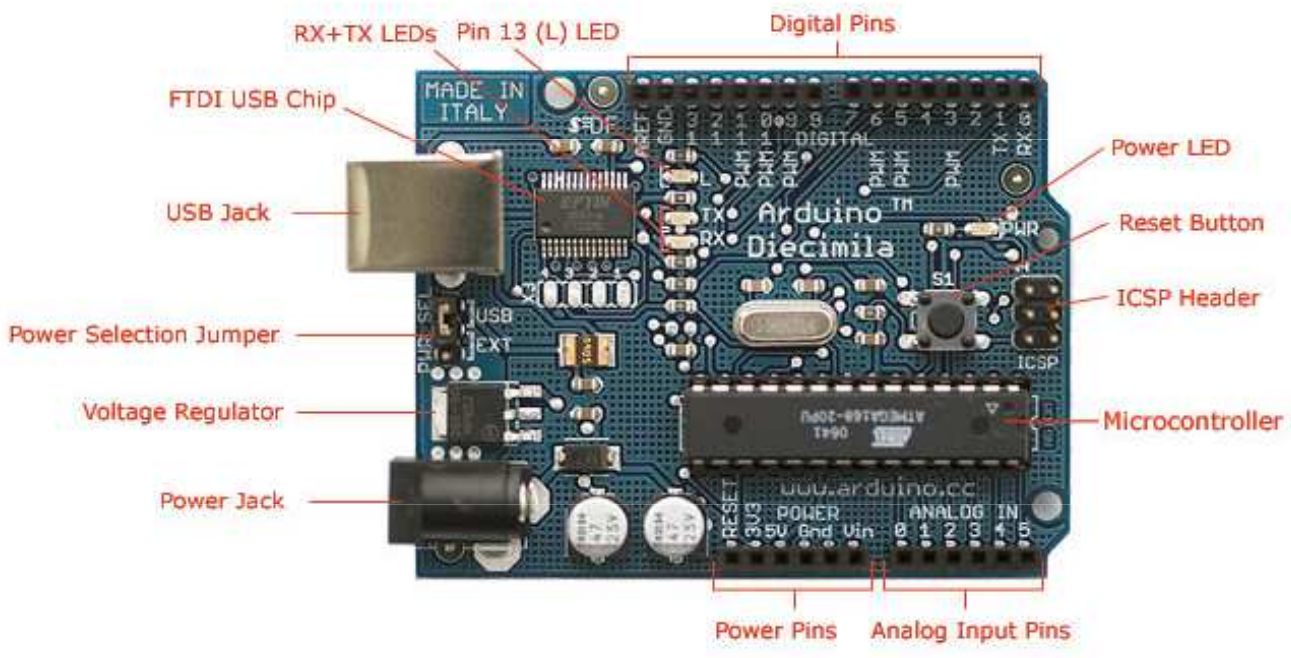


Placas Arduino compatibles



MEDIALAB PRADO

Arduino Diecimila:



Photograph by SparkFun Electronics. Used under the Creative Commons Attribution Share-Alike 3.0 license.

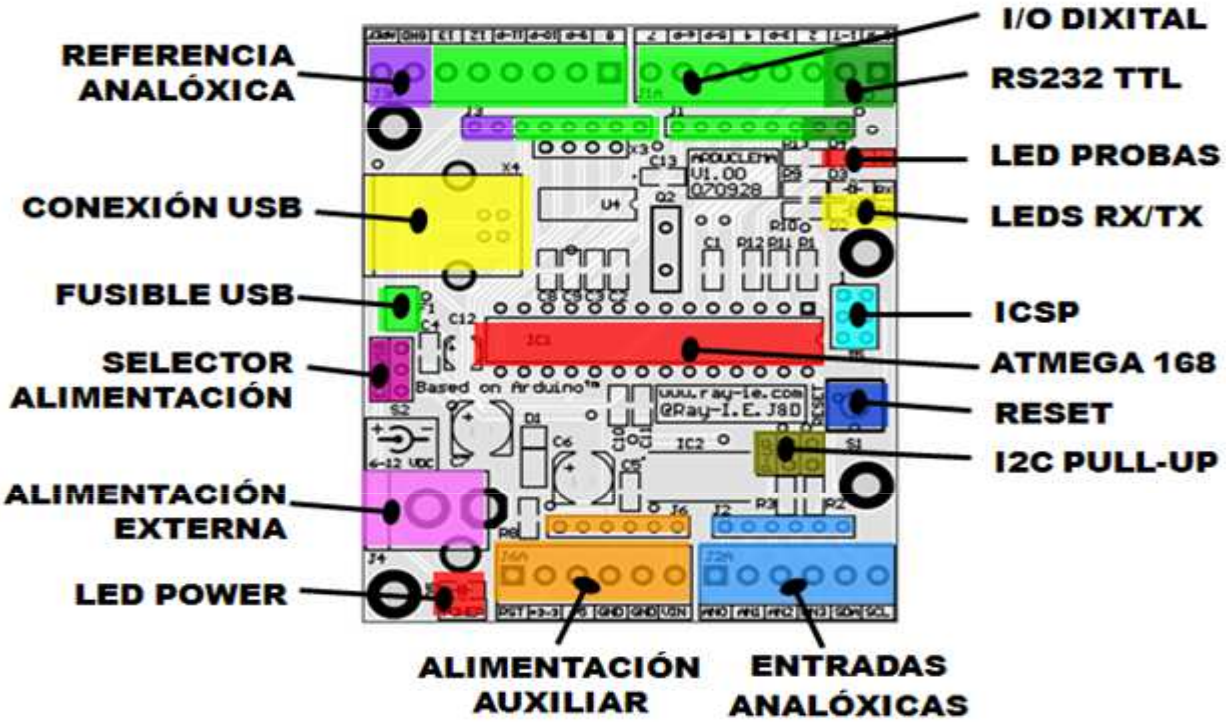
Prestacións Arduino "estándar" :

- **Conexión USB.**
- **uC Atmega 168 / 368**
- **14 E/S dixitais.**
- **6 Entradas analóxicas.**
- **6 S. analóxicas (PWM).**
- **Velocidade reloxo: 16 MHz**
- **Alimentación USB/externa.**
- **Bus I2C**

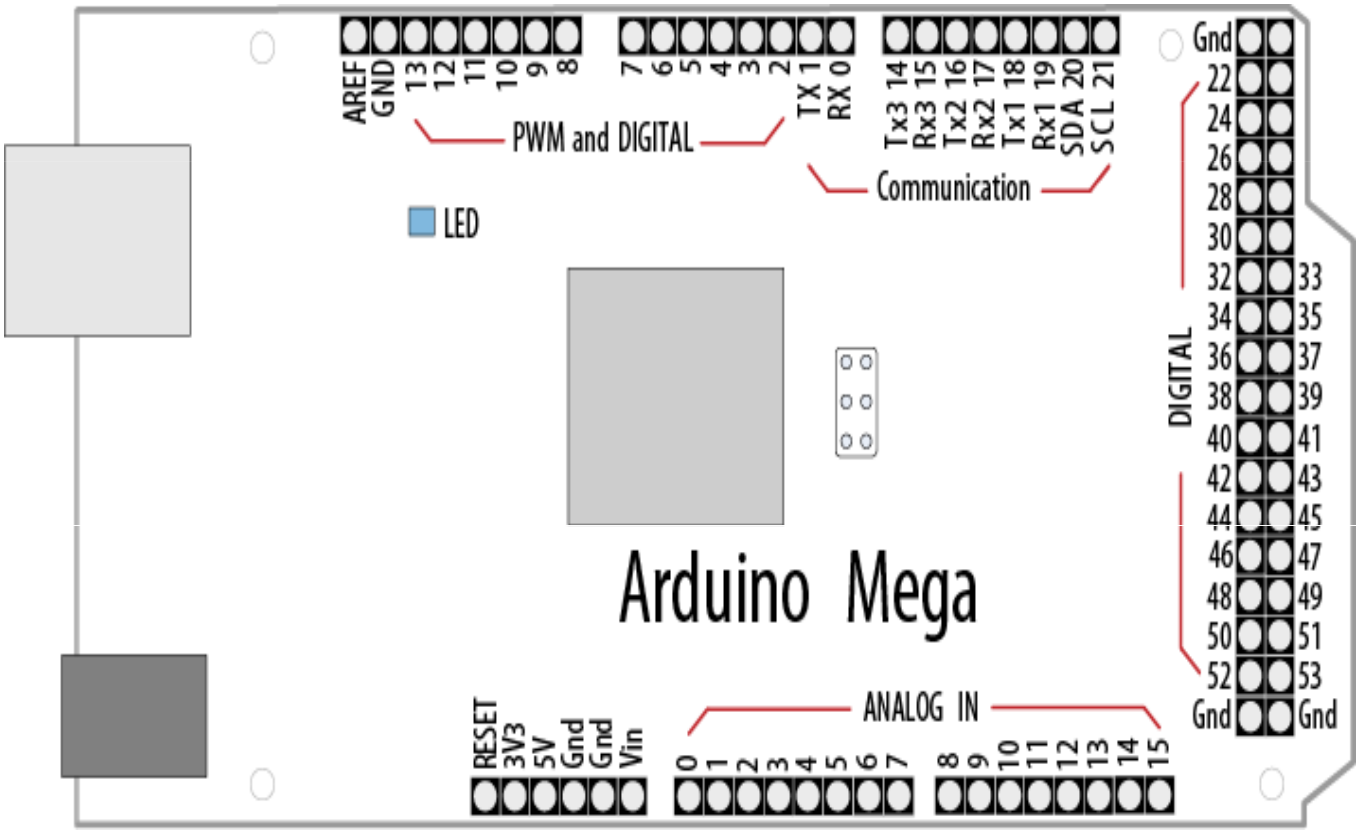
***As saídas analóxicas están compartidas coas E/S dixitais**

	Atmega 168	Atmega 328
Memoria flash *	16 KB	32 KB
RAM	1KB	2KB
EEPROM	512 B	1KB
* Bootloader Arduino	-2KB	-2KB

Versión Arduclema:



Arduino MEGA



Prestacións Arduino MEGA :

- **uC Atmega 1280/2560**
- **54 E/S dixitais.***
- **16 Entradas analóxicas.**
- **14 S. analóxicas (PWM).***
- **Velocidade reloxo: 16 MHz**

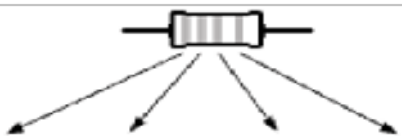
***As saídas analóxicas están compartidas coas E/S dixitais**

	Atmega 1280	Atmega 2560
Memoria flash *	128 KB	256 KB
SRAM	8 KB	8 KB
EEPROM	4 KB	4 KB
* Bootloader Arduino	-4 KB	- 4 KB

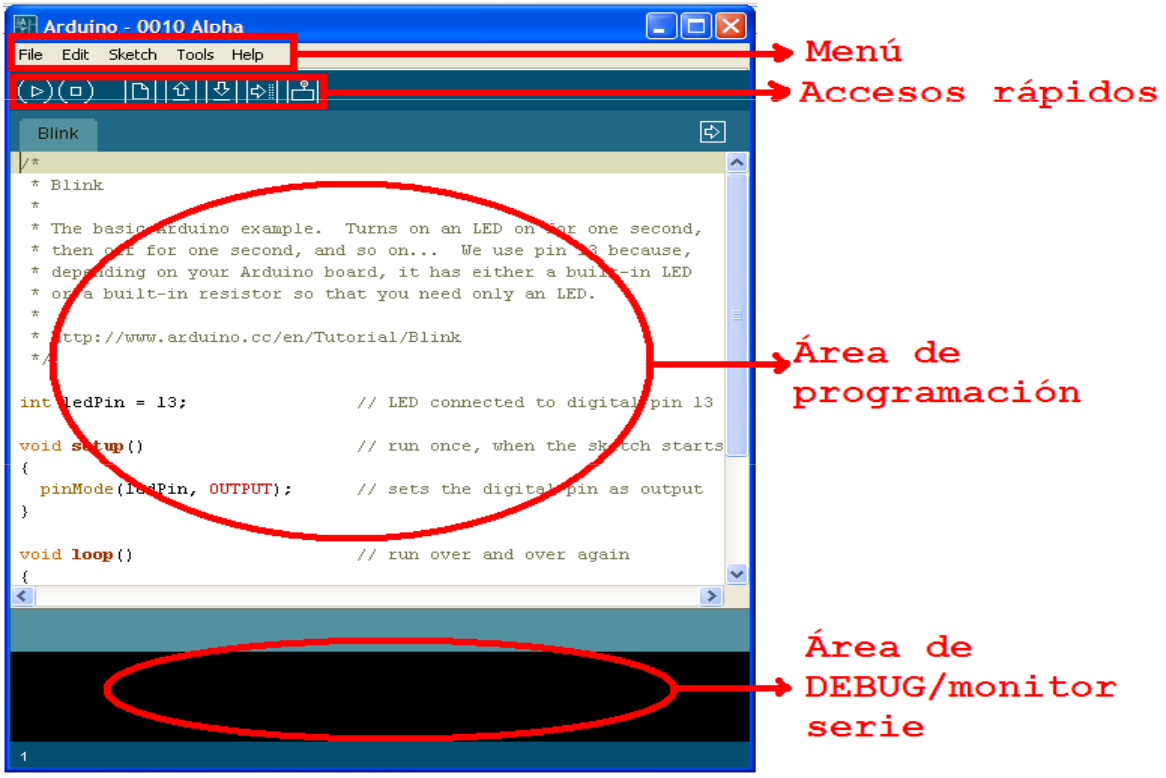
KIT PRÁCTICAS BÁSICAS

-

IDENTIFICACIÓN RESISTORES:

Código de cores				
Colores	**1ª Cifra**	**2ª Cifra**	**Multiplicador**	**Tolerancia**
Negro		0	0	
Marrón	1	1	x 10	±1%
Vermello	2	2	x 10^{2}	±2%
Laranxa	3	3	x 10^{3}	
Amarelo	4	4	x 10^{4}	
Verde	5	5	x 10^{5}	±0.5%
Azul	6	6	x 10^{6}	
Violeta	7	7	x 10^{7}	
Gris	8	8	x 10^{8}	
Branco	9	9	x 10^{9}	
Ouro			x 10^{-1}	±5%
Prata			x 10^{-2}	±10%
Sen cor				±20%

Entorno de desenvolvemento (IDE)



INSTALACIÓN:

- **Adquirir unha placa Arduino.**
- **Descargar IDE en**

http://arduino.cc/en/Main/Software

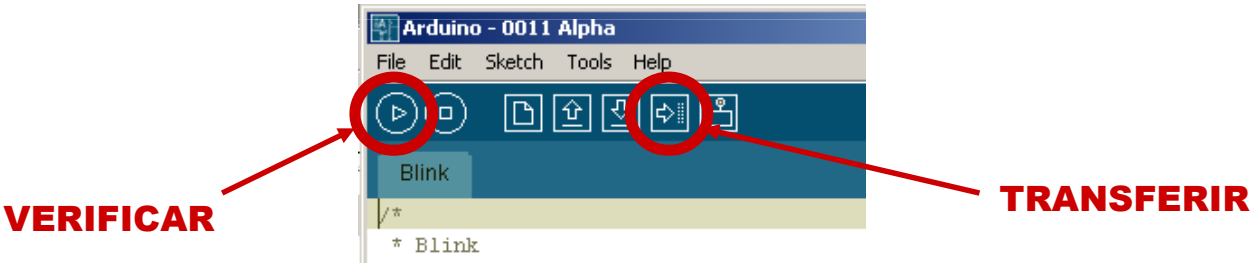
- Windows: Drivers + descomprimir e executar
- Linux: Descomprimir o paquete e executar.
- MAC: Drivers + descomprimir e executar.

*Ver anexos

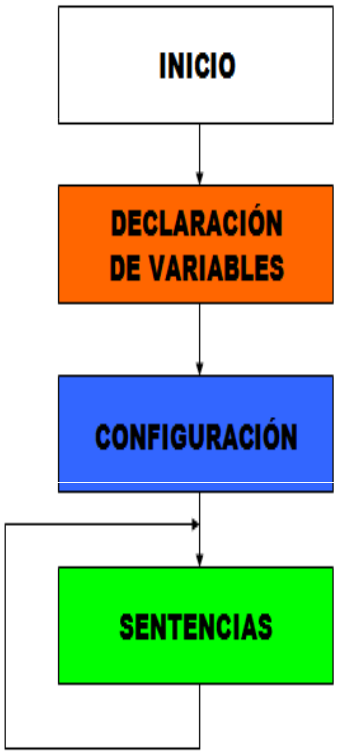
PRÁCTICA 1 - BLINK

- **DESCRIPCIÓN:**
  - **Fai pestanexar o LED de probas incluído na placa (E/S dixital 13).**

- **EXERCICIO:**
  - **Introducir o programa seguinte.**
  - **Verificar e transferir o programa.**



ESTRUTURA DO PROGRAMA




```
int ledPin = 13;                  // LED probas da placa: pin 13
                                  // Outras variables
void setup()                      // Bloque de configuración
{
pinMode(ledPin, OUTPUT);   // Fixa pin 13 como saída.
}

void loop()                       // Bloque de execución cíclica.
{
digitalWrite(ledPin, HIGH);    // Pon a saída a “1”
delay(1000);                      // Espera 1 s
digitalWrite(ledPin, LOW);     // Pon a saída a “0”
delay(1000);
}
```

PRÁCTICA 1 - BLINK

- ***REFERENCIAS:***
  - ***int* : Variables enteiras.**
  - ***setup()* : Bloque de configuración**
  - ***loop()* : Bloque de execución cíclica.**
  - **Función *pinMode()* : Configuración de pins E/S**
  - **Función *digitalWrite()* : escritura da saída dixital.**
  - **Función *delay()* : retardo en milisegundos.**

- **AMPLIACIÓN:**
  - **Cambiar tempos de pestanexo.**
  - **Observar resultados.**

Os pins poden ser utilizados como entrada ou saída dixital dixital, con excepción dos pins 0 (TX) e 1 (RX) usada para a comunicación serie ou comunicación do Arduino con outros dispositivos.

Comandos básicos:
• función setup () é a configuración de Arduino e é executado só unha vez, mentres loop () é executado de continuamente ata que se apaga o sistema.

•pinMode (pin, modo); é usado para declarar un pin dixital como entrada (input) ou saída (output). Os pins analóxicos son, por entrada estándar.

•delay (tempo); serve para deter os procesos da tarxeta durante T (ms)

•delayMicroseconds (T) ; Deter o proceso en placa T (us).

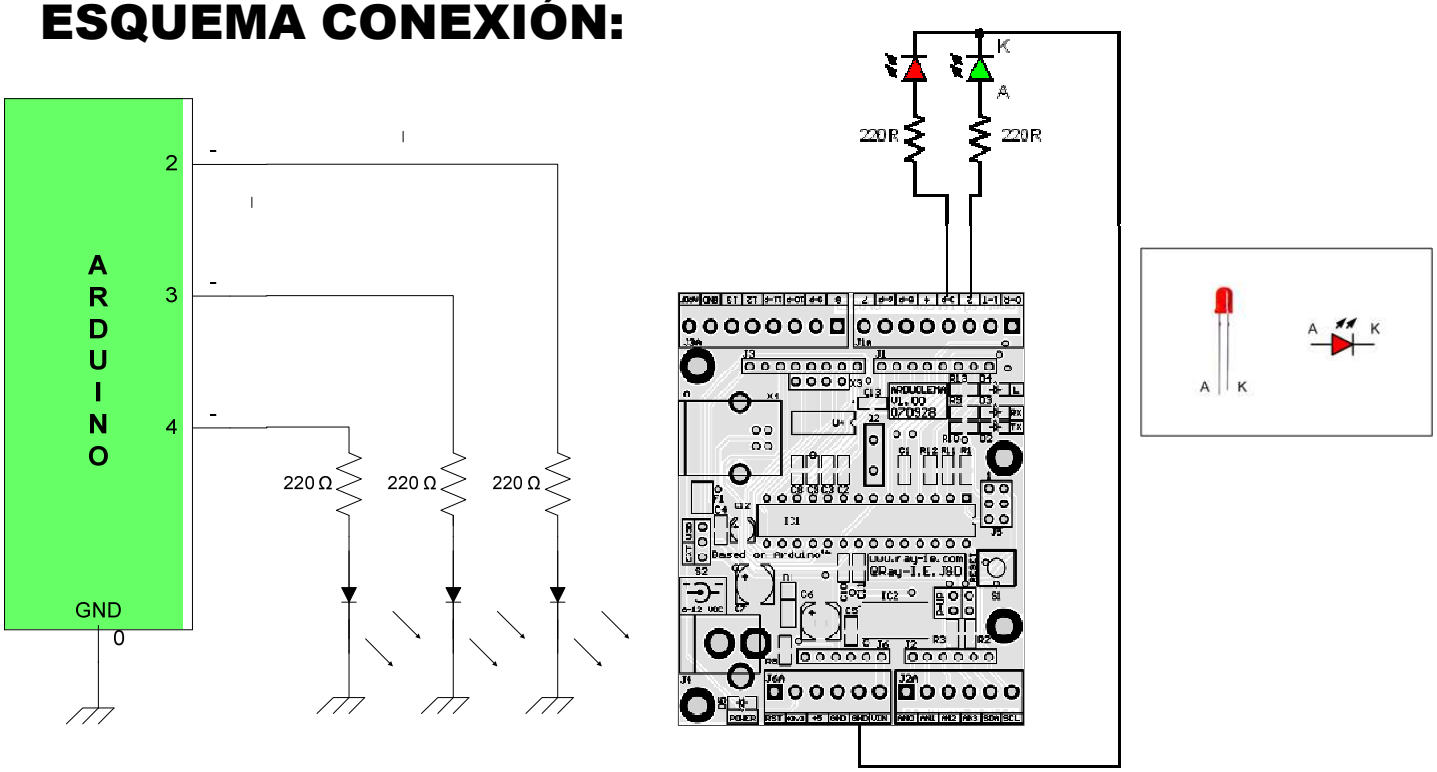
•digitalWrite (Pin, valor); Escribe un valor que pode ser (HIGH = 5V) ou (LOW = 0V)

PRÁCTICA 2 – SEMÁFORO

- **DESCRICIÓN:**
  - **Realizar un programa que simule o funcionamento dun semáforo con 3 leds.**
- **EXERCICIO:**
  - **Cablear na protoboard 3 leds como mostra o esquema.**
  - **Realizar o programa.**
- **REFERENCIAS:**
  - **As mesmas da práctica 1.**

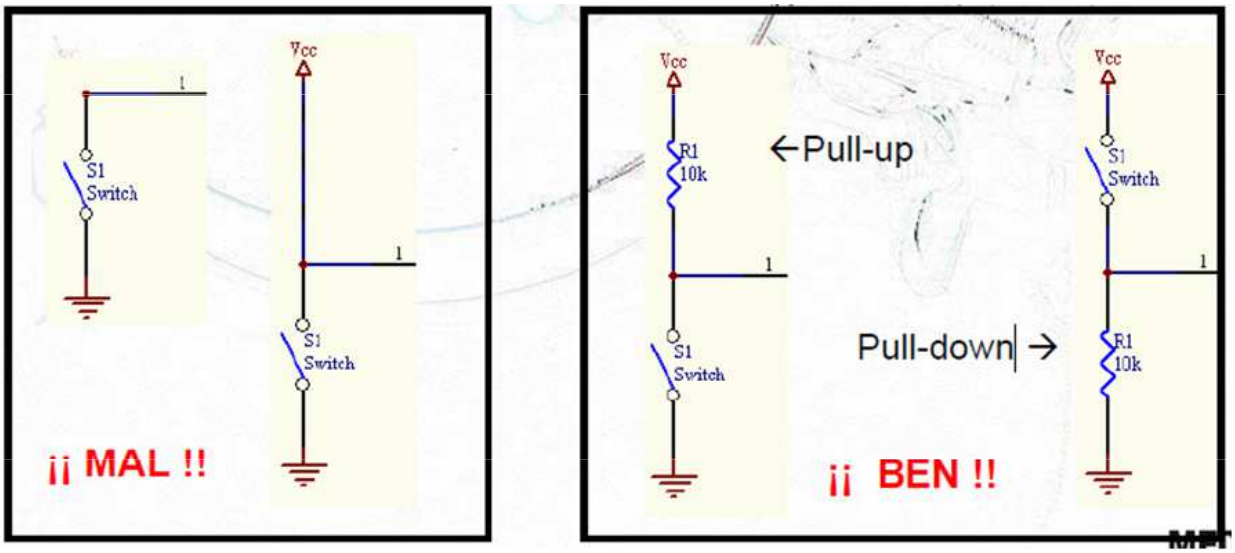
PRÁCTICA 2 – SEMÁFORO

- **ESQUEMA CONEXIÓN:**



ENTRADAS DIXITAIS

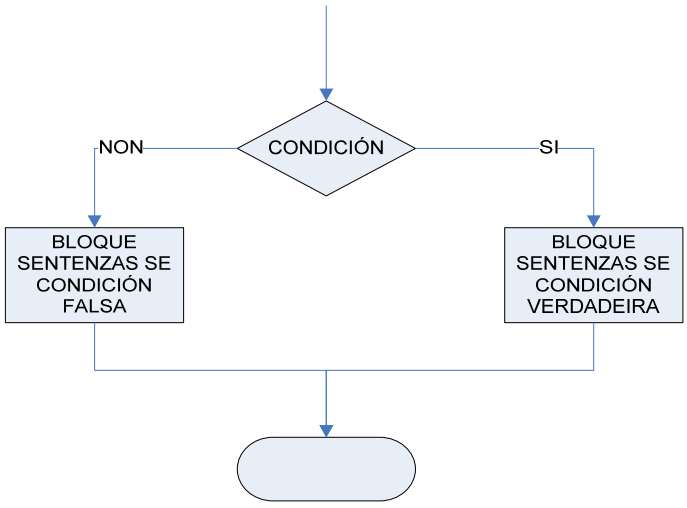
Resistencias pull-down e pull-up



Serve para ler valores dixitais sen que o valor de entrada sexa indeterminado.

ENTRADAS DIXITAIS

- ***REFERENCIAS:***
  - **Función *digitalRead()* : lectura de entrada dixital.**
  - **Función *if...else: condicións.***




```
if(Condición)
{
bloque de sentenzas se a condición é verdadeira
}
else
{
bloque de sentenzas se a condición é falsa
}
```

*** <u>Exemplo de lectura dixital:</u>**

```
int pulsPin=8;          // Na declaración

pinMode(pulsPin,INPUT);        // Na congiguración.


if(digitalRead(pulsPin) == HIGH) ….
else ….
```

PRÁCTICA 3 – ENTRADA DIXITAL

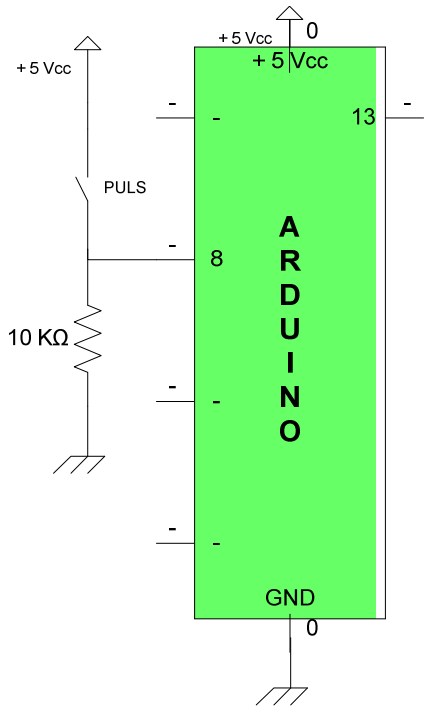
- **DESCRICIÓN:**
  - **Realizar un programa que faga alumear o led de probas ao actuar sobre un pulsador.**

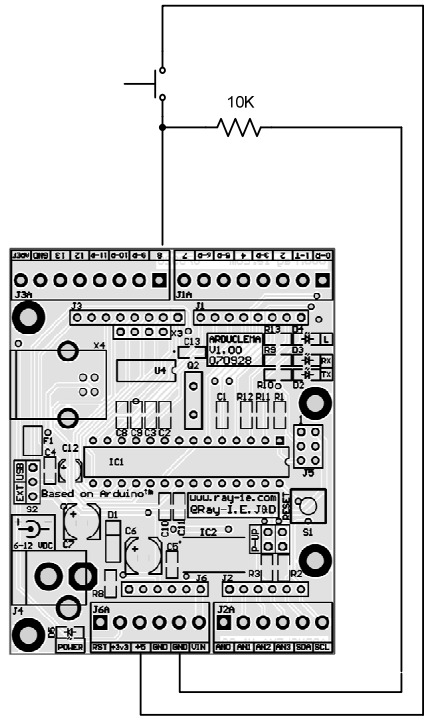
- **EXERCICIO:**
  - **Cablear 1 pulsador con pulldown como mostra o esquema.**
  - **Realizar o programa.**

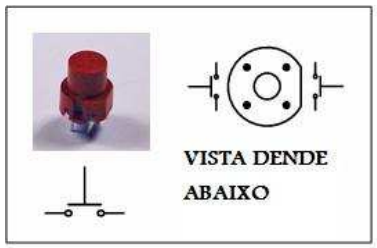
- **EXERCICIO EXTRA:**
  - **Facer que o LED permaneza encendido durante 5 s dende que se libere o pulsador.**

PRÁCTICA 3 – ENTRADA DIXITAL

- **ESQUEMA CONEXIÓN:**







MONITORIZACIÓN SERIAL

Serve para enviar datos dos sensores para o ordenador ou incluso para enviar comandos do PC para o Arduino.

Funcións de referencia:

- *Serial.begin(BAUDIOS)* : configuración comunicación serie.
- *Serial.print(DATO)*: envía dato por porto serie.
- *Serial.println(DATO)*: envía dato por porto serie seguido dun retorno de carro.
- *Activación da monitorización no IDE*:

Nota: Se a comunicación serial está activada, non se poden usar os pins 0 e 1 como entrada / saída dixital.

PRÁCTICA 4 –Contador de pulsacións

- Descrición: Facer un programa que conte o número de actuacións sobre un pulsador e visulaice un contador no ordenador.

- Esquema eléctrico : O mesmo ca no anterior.

- Exercicio extra: Facer que o led de probas da placa alumee cando o contador sexa múltiplo de 3.

PRÁCTICA 5 – RECIBE SERIAL

- Obxectivo: Recibir ordes polo porto serie

- Descrición: Facer un programa que encenda un LED ao recibir un 'H' polo porto serie e que o apague se recibe un 'L'

- Referencias:

- **Serial.available() // Chegada de datos ao porto serie**
- **caracter = Serial.read(); //Lectura caracter**

PRÁCTICA 5 – RECIBE SERIAL

```
char caracter; // variable que recibe o dato do puerto serie
int ledpin = 8; // LED conectado ao pin 13
void setup() {
  pinMode(ledpin, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  if( Serial.available() ) // se hai datos...
  {
    caracter = Serial.read(); // le e almacena o dato en 'caracteracter'
  }

  if (caracter == 'H' || caracter == 'h') digitalWrite(ledpin, HIGH);
    else if (caracter=='L' || caracter == 'l') digitalWrite(ledpin, LOW);

    delay(100); // espera 100ms
}
```

Función millis()

- **millis():**

  Devolve un valor de tipo “unigned long” co tempo (ms) dende que se iniciou o programa. A variable desborda aproximadamente nuns 50 días.

- **Problema:**

  - As temporizacións con delay() impiden que se sigan facendo outras tarefas mentres se espera.

  - Empregando a función millis() podemos relizar temporizacións sen necesidade de bloquear o proceso.

  Tutorial: http://arduino.cc/en/Tutorial/BlinkWithoutDelay

Uso de millis() para temporizacións

```
const int ledPin =  13;
unsigned long inicio=0;   //Variable para almacenar o momento no que se inicia a temporización
boolean ledState;   //Flag auxiliar para encender/apagar o LED como báscula
long temporizacion = 1000;           // intervalo de temporización (milisegundos)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  ////////////////////////////////////////////////////////////////////
  // Aquí estaría o codigo que se debe executar simultameamente//
  ////////////////////////////////////////////////////////////////////

  if( millis() - inicio > temporizacion) { //Se pasou o tempo...

    inicio = millis();    // Reiniciamos a conta

    if (ledState == LOW) ledState = HIGH;
    else ledState = LOW;    // Facemos bascular ledState

    digitalWrite(ledPin, ledState);  // Aplicamos a saída
  }
}
```

ANEXO : GUIA RÁPIDA DE LINGUAXE ARDUINO

ESTRUTURA

Estrutura principal

- void setup() (estructura de configuración)
- void loop() (estructura del bucle principal)

Estruturas de control

- if()
- if()...else
- for()
- switch()...case
- while()
- do()... while
- break
- continue
- return
- goto

Sintaxe

- ; (fin de sentencia)
- {} (corchetes)
- // (comentario liña única)
- /* */ (comentario multiliña)

Operadores aritméticos

- = (asignación)
- + (suma)
- - (resta)
- * (multiplicación)
- / (división)
- % (módulo)

Operadores de comparación

- == (igual que)
- != (distinto que)
- < (menor que)
- > (maior que)
- <= (menor ou igual que)
- >= (mayor ou igual que)

Operadores booleanos

- && (and)
- || (or)
- ! (not)

Operadores compostos

- ++ (incremento)
- -- (decremento)
- += (suma composta)
- -= (resta composta)
- *= (multiplicación composta)
- /= (división composta)

Constantes

- HIGH | LOW
- INPUT | OUTPUT
- true | false

Tipos de datos

- boolean (booleano)
- char (caracter)
- byte (byte)
- int (enteiro)
- unsigned int (enteiro sin signo)
- long (enteiro longo)
- unsigned long (enteiro longo sen signo)
- float (coma flotante)
- double (coma flotante doble)
- string (cadena de texto)
- array (array)
- void (nada)

FUNCIÓNS

Funcións I/O dixitais

- pinMode(pin, mode)
- digitalWrite(pin, value)
- int digitalRead(pin)

Funcións I/O analóxicas

- int analogRead(pin)
- analogWrite(pin, value)

Funcións I/O avanzadas

- shiftOut(dataPin, clockPin, bitOrder, value)
- unsigned long pulseIn(pin, value)

Funcións de tempo

- unsigned long millis()
- delay(milisegundos)
- delayMicroseconds(microsegundos)

Funcións matemáticas

- min(x, y)
- max(x, y)
- abs(x)
- constrain(x, a, b)
- map(value, fromLow, fromHigh, toLow, toHigh)
- pow(base, exponente)
- sq(x)
- sqrt(x)
- sin(rad)
- cos(rad)
- tan(rad)

Funcións números aleatorios

- randomSeed(semilla)
- long random(max)
- long random(min, max)

Comunicacións serie

- Serial.begin(baudios)
- int Serial.available()
- int Serial.read()
- Serial.flush()
- Serial.print(datos)
- Serial.println(datos)

LIBRERÍAS ESTÁNDAR

EEPROM – Lectura e escritura en memoria permamente.

Ethernet – Conexión a redes Ethernet

Firmata – Para comunicar con aplicacións de ordenador usandoo protocolo serie estándar

LiquidCrystal – Para controlar pantallas de cristal líquido (LCDs)

SD – Traballar con tarxetas SD

Servo – Para manexar servomotores

SPI – Comunicaicón con dispositivos con bus Serial Peripheral Interface (SPI).

SoftwareSerial – Para emular comunicacións serie con calquer pin

Stepper – Para manexar motores paso a paso

Wire - Para empregar o protocolo I2C con outros dispositivos ou sensores.