Objetivos

- Instalar el IDE Arduino en distintos sistemas operativos.
- Reconocer las distintas prestaciones del IDE.
- Comprender la sintaxis básica de Arduino y las funciones de la librería estándar.

CLASE 3

IDE Arduino

De las siguientes direcciones se puede descargar la versión de IDE que corresponda a su SO. **Arduino es multiplataforma**.

La **versión** que vamos a utilizar es la **1.0.5** que tiene fecha de release: Mayo de 2013.

- **https://arduino.googlecode.com/files/arduino-1.0.5-macosx.zip**
- **https://arduino.googlecode.com/files/arduino-1.0.5-windows.exe**
- **https://arduino.googlecode.com/files/arduino-1.0.5-linux64.tgz**
- **https://arduino.googlecode.com/files/arduino-1.0.5-linux32.tgz**

IDE Arduino

Las versiones correspondientes a Windows y MacOSx son muy estables y no suelen presentar problemas al instalarlas.

La instalación en Ubuntu u otros entornos linux suele necesitar un tiempo extra hasta quedar totalmente estable y funcional.

En el caso de optar por entornos libres se recomienda utilizar Ubundu 12.04

FTDI - IDE Arduino

Las placas Arduino UNO anteriores a la r3 poseen un chip USB FTDI a diferencia de **la r3** que trae un chip **ATmega16U2**.

Para trabajar con este chip necesitamos **instalar el driver correspondiente** que se entrega junto con el instalador.

IDE Arduino – Ubuntu 10.10

No utilizar las versiones disponibles en los repositorios ya que son obsoletas.

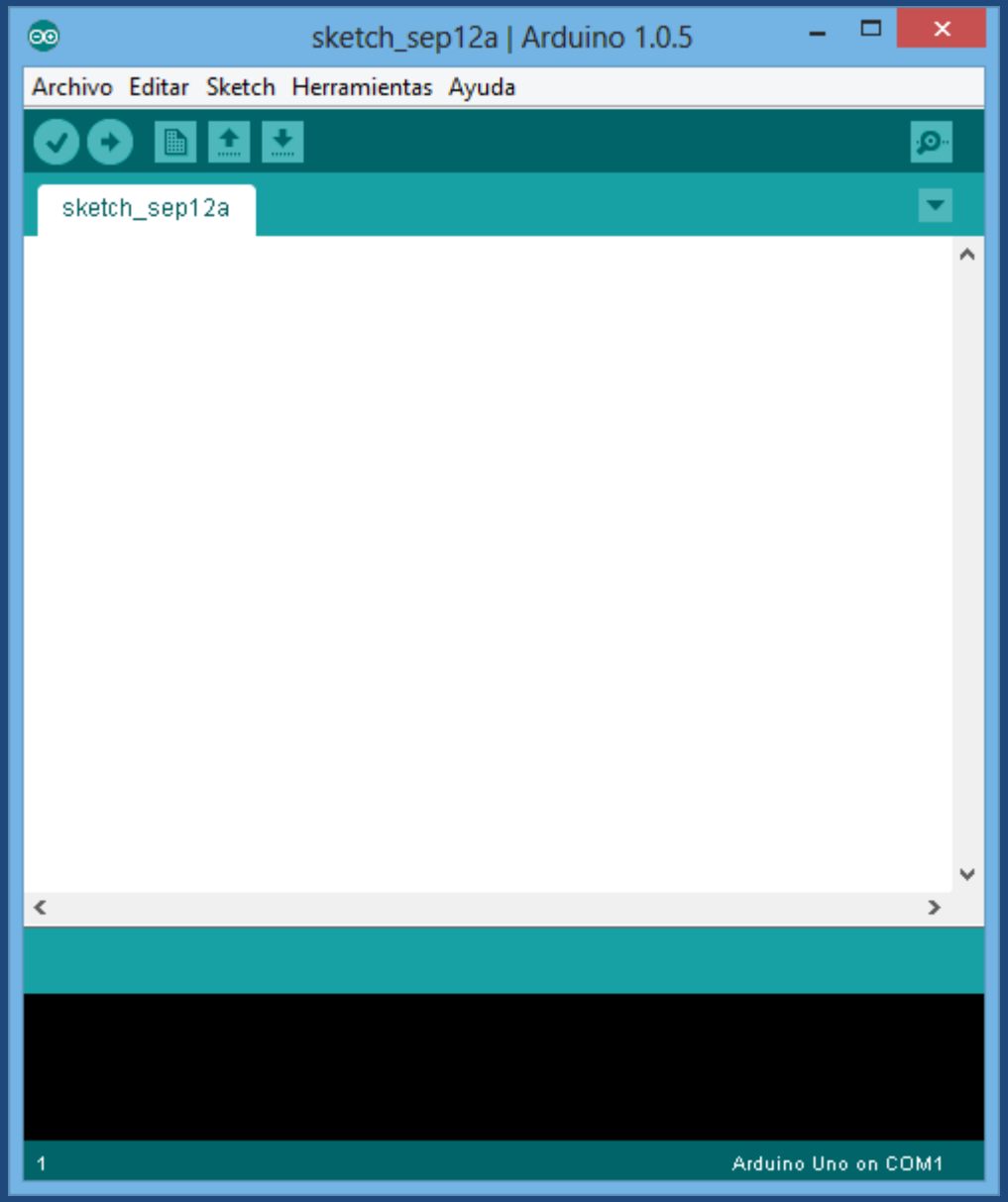
- Desde System > Administration ejecutar el Synaptic Package Manager.
- Instalar los paquetes: **Openjdk-6-jre, gcc-avr, avr-libc** y todas las dependencias que indique Ubuntu necesarias.
- Reiniciar el SO
- Descargar la versión Arduino correspondiente a su SO
- Descomprimirla y disparar el archivo ejecutable “arduino”

IDE Arduino – Ubuntu 12.04

Ejecutar desde terminal los siguientes comandos y seguir las instrucciones en pantalla:

- sudo apt-get update
- sudo apt-get install arduino arduino-core

Descargar luego de la web oficial la versión deseada correspondiente al SO, descomprimirla y ejecutar el archivo “arduino”.



Puerto
Tipo de placa
Verificación
Upload
Nuevo
Abrir
Guardar
Monitor
Consola
Estado

Sintaxis básica

Comentarios	// o /**/
Llaves	{ }
Paréntesis	( )
Separador de sentencias	;

Operadores

Aritméticos	=, +, -, *, /, %
Booleanos	==, !=, <, >, <=, >=, &&, ||
Compuestos	+=, -=, *=, /=, ++, --

Variables y constantes

Una **variable** es un **lugar reservado en memoria con un tipo de datos asociado, un nombre y un valor**.

Las **constantes** a diferencia de las **variables no cambian su valor una vez definidos**.

Las constantes **se definen anteponiendo la palabra reservada const** delante del tipo de la constante:

- const int a = 1;

Variables y constantes

El scope hace referencia a la visibilidad de un miembro.

```
Int a = 2;
void loop ()
{
        int b = 3;
        for (int c = 1; c <= 10; c++) { }
}
```

Funciones SETUP y LOOP

```
void setup ( )
{
        // única ejecución
}

void loop ( )
{
        // ejecución eterna
}
```

Estructuras de control

```
if ( evaluacion ) {
        // verdadero
} else {
        // falso
}

( evaluacion ) ? ( verdadero ) : ( falso )
```

Estructuras de control

```
switch ( valor )
{
        case A :
                // código
                break;
        case B :
                // código
                break;
        default :
                // código
                break;
}
```

Estructuras de control

```
for ( inicialización; condición; incremento)
{
        // código
}
```

Estructuras de control

```
while ( condicion )
{
        // código
}
```

Estructuras de control

```
do
{
        // código
}
while ( condicion );
```

Funciones definidas por el usuario

```
tipo_retorno nombre (
        tipo_arg_1 arg_1,
        tipo_arg_2 arg_2
)
{
        // código
}
```

Tipos escalares
numéricos

Tipo	Rango	Tamaño
byte o unsigned char	Entero entre 0 y 255	8 bits
word o unsigned int	Entero entre 0 y 65,535	8 bits
int o short	Entero entre -32,768 y 32,767	16 bits
unsigned long	Entero entre 0 y 4,294,967,295	16 bits
long	Entero entre -2,147,483,648 to 2,147,483,647	32 bits
float o double	Decimal entre -3.4028235E+38 y 3.4028235E+38	32 bits

Tipos escalares
alfanuméricos, booleanos y void

Tipo	Rango	Tamaño
char	Caracter ASCII	8 bits
string	Array de chars	n + 1 bits
boolean	True o False	8 bits
void	Sin valor de retorno	

Array

La definición de arrays puede darse de las siguientes formas :

- int a[4] = { 1, 2, 3, 4 };
- byte b[] = { 7, 8, 9, 10 };

string (array de char)

Existen varias formas de definir un string como array de char:

- char str[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
- char str[8] = "arduino";

Class

Una **clase es una plantilla**, un molde.

Una clase puede ser instanciada ‘n’ veces. Tantas como sean necesarias.

Una clase instanciada es un objeto con métodos (funciones) **y propiedades** (variables de clase).

Una clase puede contener métodos y propiedades estáticos (de clase) **que pueden ser invocados sin necesidad de instanciar la clase**.

Funciones matemáticas

min	Retorna el mínimo de dos números
max	Retorna el máximo de dos números
constrain	Indica si un número se encuentra entre un mínimo y un máximo.
pow	Eleva un número a una potencia
sqrt	Obtiene la raíz cuadrada de un número
sin, cos, tan	Seno, coseno y tangente

Números aleatorios

randomSeed, random Genera números pseudo-aleatorios

Pines

pinMode	Establece al pin como OUTPUT o INPUT
digitalWrite	Escribe LOW o HIGH al pin
digitalRead	Lee un valor LOW o HIGH del pin
analogRead	Lee un valor análogo del pin entre 0 y 1023
analogWrite	Escribe un valor entre 0 y 255 a un pin PWM

Tonos

La función **tone genera un tono de onda cuadrada sobre el pin indicado de una determinada duración o hasta que la función noTone sea invocada.**

- tone (pin, frecuencia)
- tone (pin, frecuencia, duracion)
- noTone (pin)

Tiempo y retardos

millis Retorna la cantidad de milisegundo que está corriendo Arduino.
Vuelve a 0 cada 50 días (aprox.)

micros Retorna la cantidad de microsegundos que está corriendo Arduino.

Vuelve a 0 cada 70 minutos (aprox.)

1,000 milisegundos = 1 segundo

1,000 microsegundos = 1 milisegundo

1,000,000 microsegundos = 1 segundo

Tiempo y retardos

delay Pausa el programa la cantidad de milisegundos indicada.

Serial

La clase Serial **posee métodos estáticos que permiten la comunicación entre dispositivos por medio del puerto USB** de nuestra placa Arduino.

Normalmente esto **se utiliza para hacer debug** de aplicaciones **y para interactuar con otros sistemas en una PC**.

Serial

begin	Comienza la comunicación serial a una determinada cantidad de baudios.
available	Indica si existen bytes que leer.
read	Lee el primer byte disponible en cola.
print	Imprime el texto por el canal serial en formato ASCII
println	Idem print, pero agrega /r/n al final
end	finaliza la comunicación serial y libera los pines 0 y 1

Interrupciones

Las interrupciones son **funciones que se ejecutan cuando ocurre un evento determinado sobre un pin**. **Arduino UNO r3 posee dos pines de interrupción los números 2 y 3 correspondientes a la interrupción 0 y 1**.

Interrupciones

attachInterrupt (interrupt, function, mode)

interrupt = 0 o 1 dependiendo del pin

function = nombre de la funcion a ejecutar

mode = LOW, CHANGE, RISING, FALLING

Interrupciones

detachInterrupt (interrupt)

interrupt = 0 o 1 dependiendo del pin

Interrupciones

Se pueden detener o rehabilitar las interrupciones utilizando las funciones:

- noInterrupts();
- interrupts();